HbbTV 2.0.1 Specification

2016-07-04

Contents

Introduction

The present document is the fourth revision of the HbbTV specification. It is based on the third revision with the intention of:

1) addressing gaps relative to ES 202 184 (MHEG-5 Broadcast profile) as used in the UK and TS 101 812 (DVB-MHP) as used in Italy and

2) addressing issues found since the completion of the third revision.

1 Scope

The present document defines a platform for signalling, transport and presentation of enhanced and interactive applications designed for running on hybrid terminals that include both a DVB compliant broadcast connection and a broadband connection to the internet.

The main uses of the broadcast connection are the following:

- Transmission of standard TV, radio and data services.
- Signalling of broadcast-related applications.
- Transport of broadcast-related applications and associated data.
- Transport of On Demand content for Push-services.
- Synchronization of applications and TV/radio/data services.

The main uses of the broadband connection are the following:

- Carriage of both On Demand and Live content.
- Transport of broadcast-related and broadcast-independent applications and associated data.
- Exchange of information between applications and application servers.
- Starting applications on a Companion Screen Device.
- Communicating with applications on a Companion Screen Device or a second hybrid terminal.
- Synchronizing media and applications between a hybrid terminal and a Companion Screen Device or a second hybrid terminal.

Applications are presented by an HTML/JavaScript browser.

The platform has the following characteristics:

- It is open and is not based on a single controlling authority or aggregator.
- Services and content from many different and independent providers are accessible by the same terminal.
- Standard functions of the terminal are available to all applications. Sensitive functions of the terminal are only available to trusted applications.
- Services and content may be protected.
- Broadcasted applications can be presented on terminals which are not connected to broadband. This includes both terminals which could be connected but have not yet been connected and terminals located where no broadband connectivity is available.
- Applications or services provided by a device manufacturer are outside the scope of the present document even if they use the same browser and features as described by the present document.
- Video, audio and system formats for the broadcast channel are outside the scope of the present document. Protocols for the broadcast channel are also outside the scope of the present document except for those relating to interactive applications and to synchronization.
- Applications can run on different types of terminals such as IDTVs, set-top boxes, and PVRs.
- Both broadcast-related and broadcast-independent applications are supported.

The platform combines a profile of the Open IPTV Forum specifications with a profile of the DVB specification for signalling and carriage of interactive applications and services in Hybrid Broadcast/Broadband environments. In addition, the present document defines supported media formats, minimum terminal capabilities, and the application life cycle.

The present document requires terminals to run applications signalled as conforming to the two previous revisions. This allows for smooth transitions where the previous revisions have been deployed.

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 reference document (including any amendments) applies.

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

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

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

[2] Open IPTV Forum Release 2 specification, volume 2 (V2.3): "Media Formats".

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

[3] TS 102 809 (V1.2.1): "Digital Video Broadcasting (DVB); Signalling and carriage of interactive applications and services in Hybrid Broadcast/Broadband environments".

[4] Open IPTV Forum Release 2 specification, volume 4 (V2.3): "Protocols".

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

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

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

[6] IETF RFC 2616: "Hypertext transport protocol - HTTP 1.1".

[7] IETF RFC 2818: "HTTP Over TLS".

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

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

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

[11] Void.

[12] CI Plus Forum, CI Plus Specification (V1.3.1) (2011-09): "Content Security Extensions to the Common Interface".

NOTE: Available at http://www.ci-plus.com/data/ci-plus_specification_v1.3.1.pdf.

[13] Void.

[14] TS 101 154: "Digital Video Broadcasting (DVB); Specification for the use of Video and Audio Coding in Broadcasting Applications based on the MPEG-2 Transport Stream".

[15] TS 102 366 (V1.2.1): "Digital Audio Compression (AC-3, Enhanced AC-3) Standard".

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

[17] Void.

[18] Open IPTV Forum Release 2 specification, volume 3 (V2.3): "Content Metadata".

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

[19] TS 101 162: "Digital Video Broadcasting (DVB); Allocation of identifiers and codes for Digital Video Broadcasting (DVB) systems".

[20] Void.

[21] Void.

[22] Void.

[23] W3C Recommendation (October 2004): XML Schema Part 2: "Datatypes Second Edition".

NOTE: Available at http://www.w3.org/TR/xmlschema-2/.

[24] IETF RFC 6265: "HTTP State Management Mechanism".

[25] Void.

[26] IEC 62481-2 (2007-08): "Digital living network alliance (DLNA) home networked device interoperability guidelines - Part 2: Media Formats, ed1.0".

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

[28] W3C Recommendation (July 2002): "Exclusive XML Canonicalization - Version 1.0".

NOTE: Available at http://www.w3.org/TR/xml-exc-c14n/.

[29] ISO/IEC 23009-1 (2014): "Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats".

[30] ISO/IEC 23001-7 (2012): "Information technology -- MPEG systems technologies -- Part 7: Common encryption in ISO base media file format files".

[31] Void.

[32] Void.

[33] Void.

[34] Void.

[35] Void.

[36] ES 202 184 (V2.3.1): "MHEG-5 Broadcast Profile".

[37] TS 103 205 (V1.2.1): "Digital Video Broadcasting (DVB); Extensions to the CI Plus Specification".

[38] EN 301 192 (V1.5.1): "Digital Video Broadcasting (DVB); DVB specification for data broadcasting".

[39] IETF RFC 5234: "Augmented BNF for Syntax Specifications: ABNF".

[40] IETF RFC 6455: "The WebSocket protocol".

[41] W3C Candidate Recommendation (20 September 2012): "The WebSocket API".

NOTE: Available at http://www.w3.org/TR/2012/CR-websockets-20120920/.

[42] W3C Recommendation (16 January 2014): "Cross-Origin Resource Sharing".

NOTE: Available at http://www.w3.org/TR/2014/REC-cors-20140116/.

[43] EBU - TECH 3380: "EBU TT-D Subtitling distribution format", version 1.0, January 2014.

[44] EBU - TECH 3381: "Carriage of EBU TT-D in ISOBMFF", version 1.0, October 2014.

[45] DVB Document A168: "Digital Video Broadcasting (DVB); MPEG-DASH Profile for Transport of ISO BMFF Based DVB Services over IP Based Networks".

NOTE: Available at https://www.dvb.org/standards/dvb_dash.

[46] ISO/IEC 13818-1 (2013): "Generic coding of moving pictures and associated audio information -- Part 1: Systems".

[47] DVB Bluebook A167-2 [06/2016]: "Digital Video Broadcasting (DVB); Companion Screens and Streams; Part 2: Content Identification and Media Synchronization".

NOTE: Available at http://www.dvb.org/resources/public/standards/a167-2_dvb-css_part_2_content-ID-and-media-sync_June2016.pdf

[48] Void.

[49] NIST Special Publication 800-57 Part 1-Rev 3: "Recommendation for Key Management: Part 1: General (Revision 3)".

[50] DIAL 2nd Screen protocol specification v1.7 - 19 November 2014.

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

[51] W3C Working Draft (08 October 2015): "HTML5.1 - A vocabulary and associated APIs for HTML and XHTML".

[52] ISO/IEC 14496-30 (2014): "Information technology - Coding of audio-visual objects - Part 30: Timed text and other visual overlays in ISO base media file format".

[53] Open IPTV Forum Release 1 specification, volume 5 (V1.2): "Declarative Application Environment".

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

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

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

[55] IETF RFC 4492: "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)".

[56] IETF RFC 5077: "Transport Layer Security (TLS) Session Resumption without Server-Side State".

[57] IETF RFC 5289: "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)".

[58] IETF RFC 5746: "Transport Layer Security (TLS) Renegotiation Indication Extension".

[59] IETF RFC 6066: "Transport Layer Security (TLS) Extensions: Extension Definitions".

[60] ISO 639-1 (2002): "Codes for the representation of names of languages -- Part 1: Alpha-2 code".

[61] ISO 639-2 (1998): "Codes for the representation of names of languages -- Part 2: Alpha-3 code".

[62] IETF RFC 2397: "The 'data' URL scheme".

[63] CI Plus Forum, CI Plus Specification (V1.4.1) (2015-11): "Content Security Extensions to the Common Interface".

NOTE: Available at http://www.ci-plus.com/data/ci-plus_specification_v1.4.1.pdf. When available, the latest errata version (V1.4.x) of the CI Plus Specification, as published on the CI Plus LLP Web site (http://www.ci-plus.com), replaces this reference.

[64] IETF RFC 4122: "A Universally Unique IDentifier (UUID) URN Namespace"

[65] W3C Working Draft (10th October 2013): "Web Audio API"

[66] W3C Working Draft (04 February 2016) : "Encrypted Media Extensions"

NOTE: Available at https://www.w3.org/TR/2016/WD-encrypted-media-20160204/

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 reference 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] Void.

[i.2] ES 202 130 (V2.1.2): "Human Factors (HF); User Interfaces; Character repertoires, orderings and assignments to the 12-key telephone keypad (for European languages and other languages used in Europe)".

[i.3] TS 101 231 (V1.3.1): "Television systems; Register of Country and Network Identification (CNI), Video Programming System (VPS) codes and Application codes for Teletext based systems".

[i.4] W3C draft in development: "How to Add a Favicon to your Site".

NOTE: Available at http://www.w3.org/2005/10/howto-favicon.

[i.5] Void.

[i.6] Open IPTV Forum Release 2.3 specification volume 5a (V2.3): "Web Standards TV Profile".

[i.7] HDMI 2.0 specification.

NOTE: An introduction to the HDMI 2.0 specification is provided at http://www.hdmi.org/manufacturer/hdmi_2_0/. The specification itself is not publicly available.

[i.8] The DIAL Application Name Registry.

NOTE: Available at http://www.dial-multiscreen.org/dial-registry/namespace-database.

[i.9] W3C Last Call Working Draft 24 April 2014: "Tracking Preference Expression (DNT)".

NOTE: Available from http://www.w3.org/TR/2014/WD-tracking-dnt-20140424/.

[i.10] IETF draft-zyp-json-schema-04 (January 2013): "JSON Schema: core definitions and terminology", F.Galiegue, K. Zyp and G. Court.

NOTE: Available from http://tools.ietf.org/id/draft-zyp-json-schema-04.txt.

[i.11] Void.

[i.12] ISO/IEC 13818-6: "Information technology -- Generic coding of moving pictures and associated audio information -- Part 6: Extensions for DSM-CC".

[i.13] "ots: Sanitiser for OpenType - Design Document"

NOTE: Available from: https://code.google.com/p/ots/wiki/DesignDoc

[i.14] POSSIBLE Mobile: “The Developer’s Guide to Unique Identifiers”

NOTE: available from https://possiblemobile.com/2013/04/unique-identifiers/

[i.15] Marketing Land: “Google Replacing “Android ID” With “Advertising ID” Similar To Apple’s IDFA”

NOTE: available from http://marketingland.com/google-replacing-android-id-with-advertising-id-similar-to-apples-idfa-63636

[i.16] WHATWG "Compatibility"

NOTE: Available at https://compat.spec.whatwg.org/

[i.17] CA/Browser Forum, (Version 1.3.3, February 4 2016): “Baseline Requirements Certificate Policy for the Issuance and Management of Publicly-Trusted Certificates”

NOTE: Available at https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-1.3.3.pdf

[i.18] W3C Candidate Recommendation, (08 October 2015): “Mixed Content”

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

[i. 19] DASH Industry Forum:(V3.2) “Guidelines for Implementation: DASH-IF Interoperability Points”

NOTE: Available from http://dashif.org/wp-content/uploads/2015/12/DASH-IF-IOP-v3.2.pdf

3 Definitions and abbreviations

3.1 Definitions

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

application data: set of files comprising an application, including HTML, JavaScript, CSS and non-streamed multimedia files

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

broadcast: classical uni-directional MPEG-2 transport stream based broadcast such as DVB-T, DVB-S or DVB-C

broadcast-independent application: interactive application not related to any broadcast channel or other broadcast data

broadcast-related application: interactive application associated with a broadcast television, radio or data channel, or content within such a channel

broadcast-related autostart application: broadcast-related application intended to be offered to the end user immediately after changing to the channel or after it is newly signalled on the current channel

NOTE: These applications are often referred to as "red button" applications in the industry, regardless of how they are actually started by the end user.

companion screen: device (not another HbbTV® terminal) that can run applications that in turn link to and work with an HbbTV® terminal or HbbTV® application. For example, a mobile phone or tablet

companion screen application: application running on a Companion Screen and either provided by a terminal manufacturer for linking to and work with the terminal (possibly including non-HbbTV® features) or provided by a service provider that can work in conjunction with an HbbTV® application running on the terminal

companion screen device: companion screen

digital teletext application: broadcast-related application which is intended to replace classical analogue teletext services

Hybrid broadcast broadband TV application: application conformant to TS 102 796 that is intended to be presented on a terminal conformant with TS 102 796. Such an application can be either a broadcast-related application or a broadcast-independent application and can transition between these application models

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

launcher application: role taken by an application that executes on a Companion Screen and provides services to support the functioning of Companion Screen applications

linear A/V content: broadcast A/V content intended to be viewed in real time by the user

non-linear A/V content: A/V content that which does not have to be consumed linearly from beginning to end for example, A/V content streaming on demand

persistent download: non-real time downloading of an entire content item to the terminal for later playback

NOTE: Persistent download and streaming are different even where both use the same protocol - HTTP. See clause 10.2.3.2.

progressive download: variant of persistent download where playback of the content item can start before the download of the content item has completed

NOTE: Progressive download is referred to as playable download in the OIPF DAE specification [1].

synchronization timeline: timeline used in communication between a Synchronization Client and the MSAS to give the Synchronization Client an understanding of the progress of time along that timeline

terminal specific applications: applications provided by the terminal manufacturer, for example device navigation, set-up or an internet TV portal

3.2 Abbreviations

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

A/V	Audio Video
AD	Audio Description
AIT	Application Information Table
API	Application Programming Interface
AVC	Advanced Video Coding
CAM	Conditional Access Module
CAS	Conditional Access System
CDN	Content Delivery Network
CEA	Consumer Electronics Association
CE-HTML	Consumer Electronics - Hypertext Markup Language
CENC	Common Encryption
CI	Common Interface
CICAM	Common Interface Conditional Access Module
CIS	Correlation Information Server
CRC	Cyclic Redundancy Check
CS	Companion Screen
CSP	Content and Service Protection
CSS	Cascading Style Sheets
CSS-CII	Interface for Content Identification and other Information
CSS-TS	Interface for Timeline Synchronization
CSS-WC	Interface for Wall Clock
CTR	Counter
DAE	Declarative Application Environment
DASH	Dynamic Adaptive Streaming over HTTP
DLNA	Digital Living Network Alliance
DOM	Document Object Model
DRM	Digital Rights Management
DSM-CC	Digital Storage Media - Command and Control
DTD	Document Type Definition
DVB	Digital Video Broadcasting

DVB-C	Digital Video Broadcasting - Cable
DVB-S	Digital Video Broadcasting - Satellite
DVB-SI	DVB Service Information
DVB-T	Digital Video Broadcasting - Terrestrial
EIT p/f	EIT present/following
EIT	Event Information Table
EPG	Electronic Program Guide
FDP	File Delivery Protocol
FEC	Forward Error Correction
FQDN	Fully Qualified Domain Name
FSA	File System Acceleration
GIF	Graphics Interchange Format
HbbTV®	Hybrid broadcast broadband TV
HEAAC	High Efficiency AAC
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol - Secure
IP	Internet Protocol
ISO	International Organization for Standardization
ISOBMFF	ISO Base Media File Format
JPEG	Joint Photographic Experts Group
KID	Key Identifier
LCN	Logical Channel Number
MMI	Man Machine Interface
MPD	Media Presentation Description
MPEG	Motion Picture Experts Group
MRS	Material Resolution Server
MSAS	Media Synchronization Application Server
OIPF	Open IPTV Forum
OITF	Open IPTV Terminal Function
PID	Packet IDentifier
PMT	Program Map Table
PNG	Portable Network Graphics
PSI	Program Specific Information
PVR	Personal Video Recorder
REST	Representational State Transfer
RCU	Remote Control Unit
SC	Synchronization Client
SD&S	Service Discovery and Selection
SDT	Service Description Table
SPTS	Single Program Transport Stream
SSL	Secure Sockets Layer
STC	System Time Clock
SVG	Scalable Vector Graphics
TLS	Transport Layer Security
TV	Television
UI	User Interface
UPnP	Universal Plug and Play
URL	Uniform Resource Locator
UTF-8	UCS Transformation Format—8-bit
XHTML	Extensible HyperText Markup Language
XML	eXtensible Markup Language

4 Overview

4.1 Applications

The web-based Hybrid Broadcast Broadband terminal as defined in the present document provides download and execution of applications which are defined as a collection of documents constituting a self-contained enhanced or interactive service. The documents of an application are HTML, JavaScript, CSS, XML and multimedia files.

The system architecture which allows for the provision of applications comprises a browser, application signalling via broadcast, broadband and from a Companion Screen Device, application transport via broadcast, broadband and from a CICAM, and synchronization of applications and broadcast services (see clause 4.2 for details).

The present document addresses the following types of application:

- Broadcast-independent application (i.e. not associated with any broadcast service). This type of application is downloaded via broadband and accesses all of its associated data via broadband.
  - Examples of this type of service are catch-up services and games where the application does not need access to any broadcast resources.
- Broadcast-related application (i.e. associated with one or more broadcast services or one or more broadcast events within a service) that may be launched automatically ("autostart") or explicitly upon user request. This type of application may be downloaded via broadband, broadcast or CICAM and may access its data via any method.
  - Examples of this type of service are electronic program guides and teletext-like services where the application may wish to present the broadcast video in a window and access other broadcast resources (e.g. EIT metadata).

The following possible uses of the browser environment are outside the scope of the present document:

- Service provider related applications as defined in the OIPF DAE specification [1].
- Using the browser environment to provide terminal specific applications such as a channel navigator or a device setup menu.
- Using the browser environment to display open Internet websites.
- Using the browser environment to support other specifications such as DLNA Remote UI or the full set of Open IPTV Forum specifications.

4.2 Architecture (informative)

4.2.1 Introduction

This clause gives an overview of the system architecture and explains the necessary functional components inside a hybrid terminal. The level of detail of this explanation is general and abstract. Details about the internal structure of the components (e.g. whether the DSM-CC client has an integrated cache or not) or about their practical implementation (e.g. whether a specific component is solved in hardware or software) are omitted. Also in practice several components could be combined in one component (e.g. a browser with an integrated application manager). The primary intention of this clause is to provide an introduction and an understanding of the overall concept and the needed components. The communication between these components is outside the scope of the present document.

4.2.2 System overview

A hybrid terminal has the capability to be connected to two networks in parallel. On the one side it can be connected to a broadcast DVB network (e.g. DVB-T, DVB-S or DVB-C). Via this broadcast connection the hybrid terminal can receive standard broadcast A/V (i.e. linear A/V content), non-realtime A/V content, application data and application signalling information. Even if the terminal is not connected to broadband, its connection to the broadcast network allows it to receive broadcast-related applications. In addition, signalling of stream events to an application is possible via the broadcast network.

In addition the hybrid terminal can be connected to the Internet via a broadband interface. This allows bi-directional communication with the application provider. Over this interface the terminal can receive application data and non-linear A/V content (e.g. A/V content streaming on demand). The hybrid terminal may also support non-real time download of A/V content over this interface. The broadband interface may also connect to Companion Screen Devices or other HbbTV® terminals on the same local network as the hybrid terminal.

Figure 1 depicts the system overview with a hybrid terminal with DVB-S as the example of the broadcast connection.

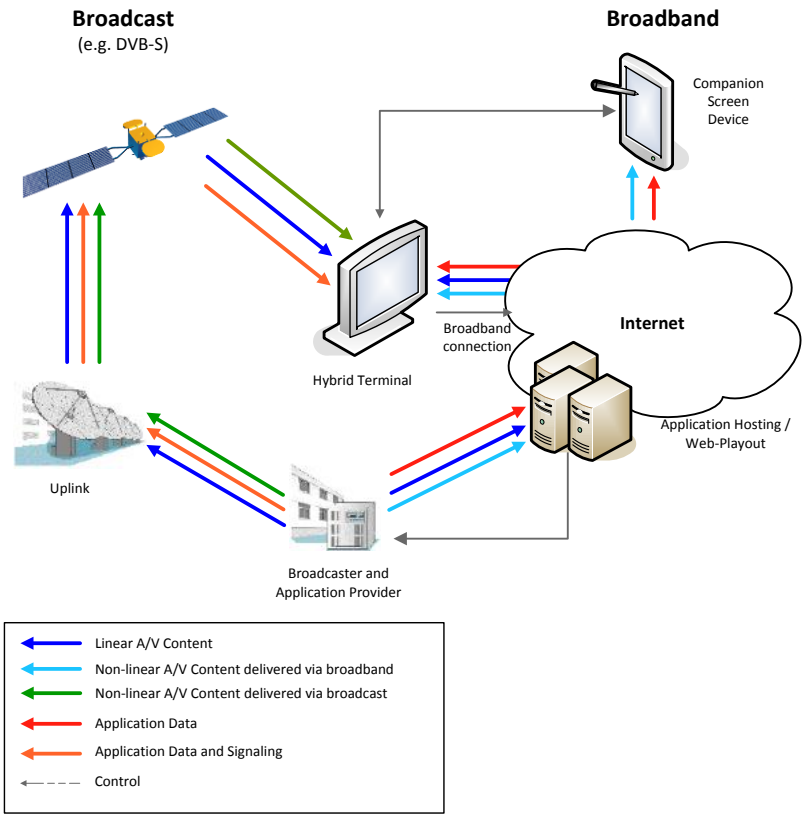


Figure 1: System Overview

4.2.3 Functional terminal components

Figure 2 depicts an overview of the relevant functional components inside of a hybrid terminal. These components are described below the figure.

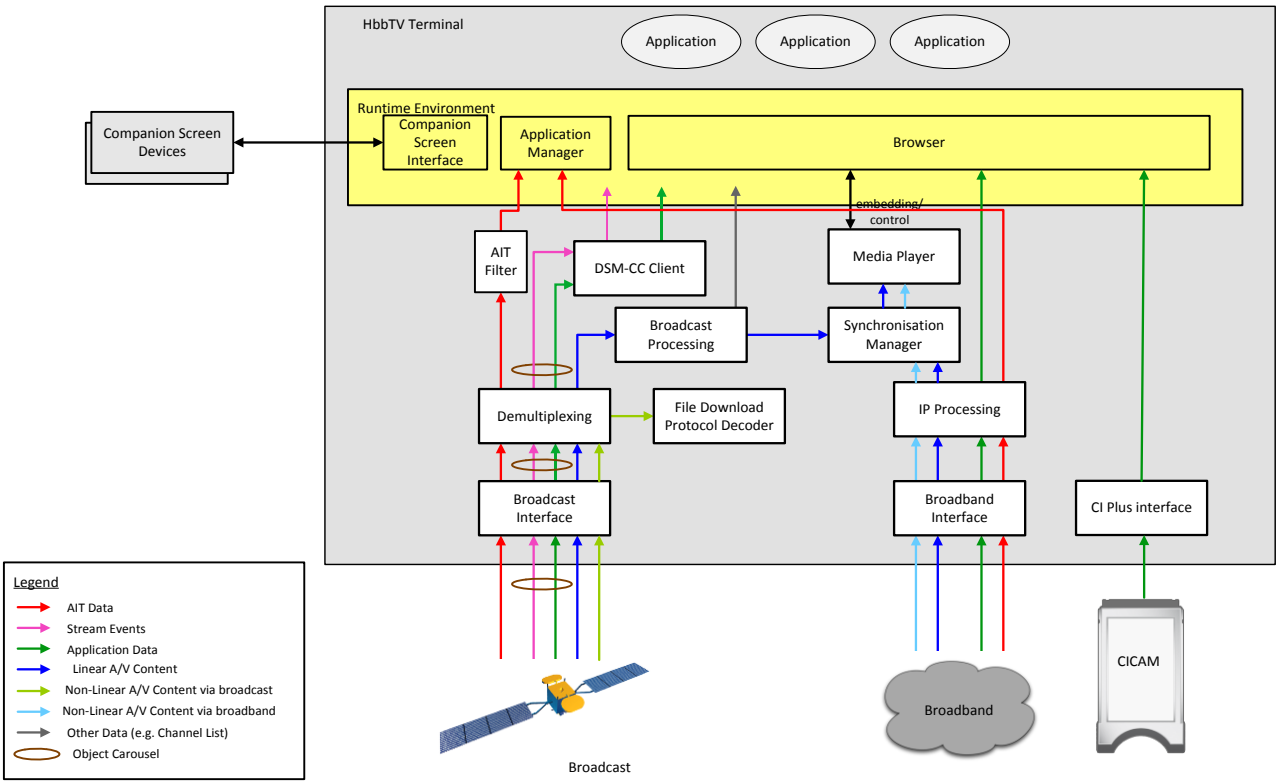


Figure 2: Functional components of a hybrid terminal

Via the **Broadcast Interface** the terminal receives AIT data, linear A/V content, non-realtime A/V content, application data and stream events. The last two data streams are transferred by using a DSM-CC object carousel. Non-realtime content is transferred by using the FDP protocol. Therefore both a **DSM-CC Client** and a **File Download Protocol Decoder** are needed to recover the data from the object carousel and FDP data stream, respectively. The recovered data is provided to the Runtime Environment. The **Runtime Environment** can be seen as a very abstract component where the interactive application is presented and executed. The Browser, an Application Manager and the Companion Screen Interface form this Runtime Environment. The **Application Manager** evaluates the AIT to control the lifecycle for an interactive application. The **Browser** is responsible for presenting and executing an interactive application.

Linear A/V content is processed in the same way as on a standard non-hybrid DVB terminal. This is included in the functional component named **Broadcast Processing** which includes all DVB functionalities provided on a common non-hybrid DVB terminal. Additionally some information and functions from the Broadcast Processing component can be accessed by the Runtime Environment (e.g. channel list information, EIT p/f, functions for tuning). These are included in the "other data" in figure 2. Moreover an application can scale and embed linear A/V content in the user interface provided by an application. These functionalities are provided by the **Media Player.** In figure 2 this includes all functionalities related to processing A/V content.

Via the **Broadband Interface** the hybrid terminal has a connection to the Internet. This connection provides a second way to request application data from the servers of an application provider. Also this connection is used to receive A/V content (e.g. for Content on Demand applications). The component **Internet Protocol Processing** comprises all the functionalities provided by the terminal to handle data coming from the Internet. Through this component application data is provided to the Runtime Environment. A/V content is forwarded to the Media Player which in turn can be controlled by the Runtime Environment and hence can be embedded into the user interface provided by an application. In combination with the **Media Player**, the **Synchronization Manager** can synchronize content delivered to the hybrid terminal via the **Broadband Interface** and content delivered to the hybrid terminal via either the **Broadband Interface** or the **Broadcast Interface**.

The **Companion Screen Interface** enables the hybrid terminal to discover **Companion Screen Devices** and other hybrid terminals and to be discovered by **Companion Screen Devices**. Through it, interactive applications running in the **Browser** can request an application be installed or started on a **Companion Screen Device** and an application running on a **Companion Screen Device** a can request the **Browser** to start an interactive application. It provides a WebSocket server to enable an interactive application in the hybrid terminal and an interactive application on either a **Companion Screen Device** or a different hybrid terminal to communicate. In combination, the **Companion Screen Interface** and the **Media Player** together enable synchronization of content delivered to the hybrid terminal via either interface with content delivered to a **Companion Screen Device** or another hybrid terminal.

Via the **CI Plus interface**, the hybrid terminal requests application data from the Auxiliary File System offered by the **CICAM**.

4.3 Terminal capabilities and extensions

The present document defines a base level (or set of capabilities for terminals) which shall be supported in all terminals. This base level supports interactive applications:

- Which do not use video as part of their UI.
- Which use broadcast video as part of their UI.
- Which use unicast streaming content on demand as part of their UI.

In addition to this base level, the present document includes four other features which may optionally be supported by terminals:

- Support for downloading A/V content from the broadcast and broadband channels into persistent memory available locally to the terminal (both persistent download and progressive download) - this is referred to as the "download feature".
- Support for scheduling and playback of recordings and time shifting of broadcast content using mass storage available locally to the terminal - this is referred to as the "PVR feature".
- Support for protected content via broadband as defined in annex B.
- Launching applications on a Companion Screen Device.

Additionally the present document defines some aspects that are mandatory for terminals supporting CI Plus [12] in whole or in part.

4.4 Specification overview

The present document specifies the technical requirements for the system described in the previous clauses. It largely references parts of already available standards and specifications and adapts these parts where necessary. The most significant referenced documents are the following:

- W3C HTML5 [54], as profiled by the OIPF Web Standards TV Profile [i.8].
- OIPF DAE specification - formally known as Open IPTV Forum Release 2 Volume 5 [1].
- DVB hybrid application signalling specification - formally known as TS 102 809 [3].
- MPEG DASH - formally known as ISO/IEC 23009-1 [29].
- MPEG CENC - formally known as ISO/IEC 23001-7 [30].

Figure 3 shows a graphical overview of the relationship between the profile defined here and the above mentioned specifications.

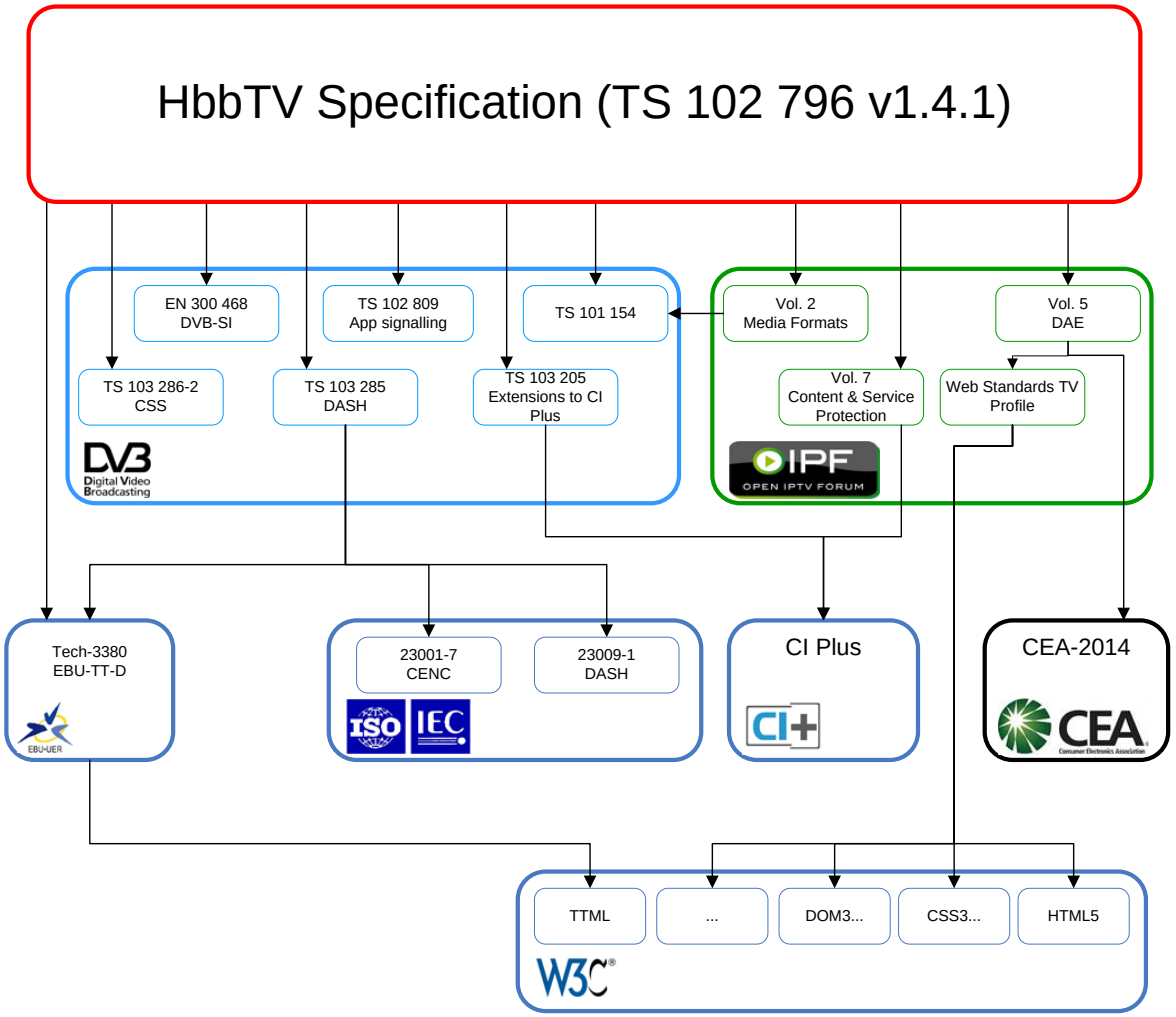


Figure 3: Specification overview

Important components provided by HTML5 [54] include:

- The HTML markup language itself.
- The `<video>` element for presenting broadband delivered video in an HTML page.
- The APIs for manipulating the contents of an HTML page from JavaScript.

Important components provided by the OIPF DAE specification [1] include:

- JavaScript APIs for applications running in a TV environment (e.g. broadcast video presentation, channel change).
- Definition of embedding linear A/V content in an application.
- Integration with content protection / DRM technologies

TS 102 809 [3] provides the following components:

- Application signalling.
- Application transport via broadcast or HTTP.

The audio and video formats are defined in the OIPF Media Formats specification [2].

In some rare cases none of the referenced standards provide an appropriate solution. In these cases the requirements are directly defined in the present document (e.g. the application lifecycle definition). Additionally the present document provides recommendations on the user experience and a description of the system overview.

The requirements in the OIPF and DVB specifications are only included if explicitly referenced in the present document or a dependency of those explicitly referenced parts. Other parts of those specifications are not required by the present document and should not be implemented unless required by another specification.

4.5 Referenced W3C Specifications

The present document requires support for the OIPF DAE Web Standards TV Profile (see clause A.1). Some normative references in that document are to W3C specifications which have not yet been fully approved or which might still be under development. In these cases, the reference is to a specific dated version of the specification. Terminals shall either implement the indicated parts of the referenced W3C specification version or the equivalent parts of a later published version of the specification which is either a Working Draft, a Candidate Recommendation, a Proposed Recommendation or a W3C Recommendation. These versions may be found by using the "This version", "Latest version" and "Previous version" links on the referenced web page for the W3C specification.

Some features in web specifications are commonly implemented with vendor prefixes, either as well as the unprefixed version or instead of the unprefixed version. For more information see the WHATWG compatibility document [i.16].

For features required by the present document or normative references from the present document, HbbTV terminals shall always support the unprefixed version and may support the prefixed version. See also A.3.12 for requirements on HbbTV applications.

5 User experience (informative)

5.0 Introduction

This clause describes the behaviour of the terminal as seen by the end-user. It should be considered as usability guidelines for implementing interactivity. However, the described behaviour usually results from the functionality coded into the broadcast application, rather than the terminal.

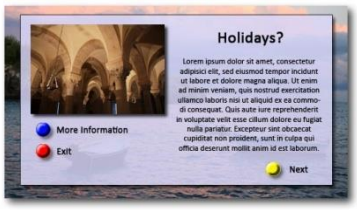
A homogenous user experience is important to enable a successful interactive platform. To ensure this, both the manufacturer and the application developer should respect the following framework and guidelines.

5.1 Visual appearance of interactive applications

5.1.1 Balance of video and application

Table 1 illustrates the range of different visual appearances the end user might experience. Each "screen" shows a different balance between "conventional TV" content and information delivered by an interactive application.

Table 1: Typical range of programme types perceived by end users

	1. Conventional TV
	2. TV with visual prompt of available information ("Red Button")
	3. TV with information overlaid (still picture only in the overlaid information, no A/V in overlay)
	4. Information with video, audio or picture inset
	5. Just information (without A/V)

5.1.2 Service selection and event change

The end-user may see a change in appearance either when she/he changes channel or when a service changes through time.

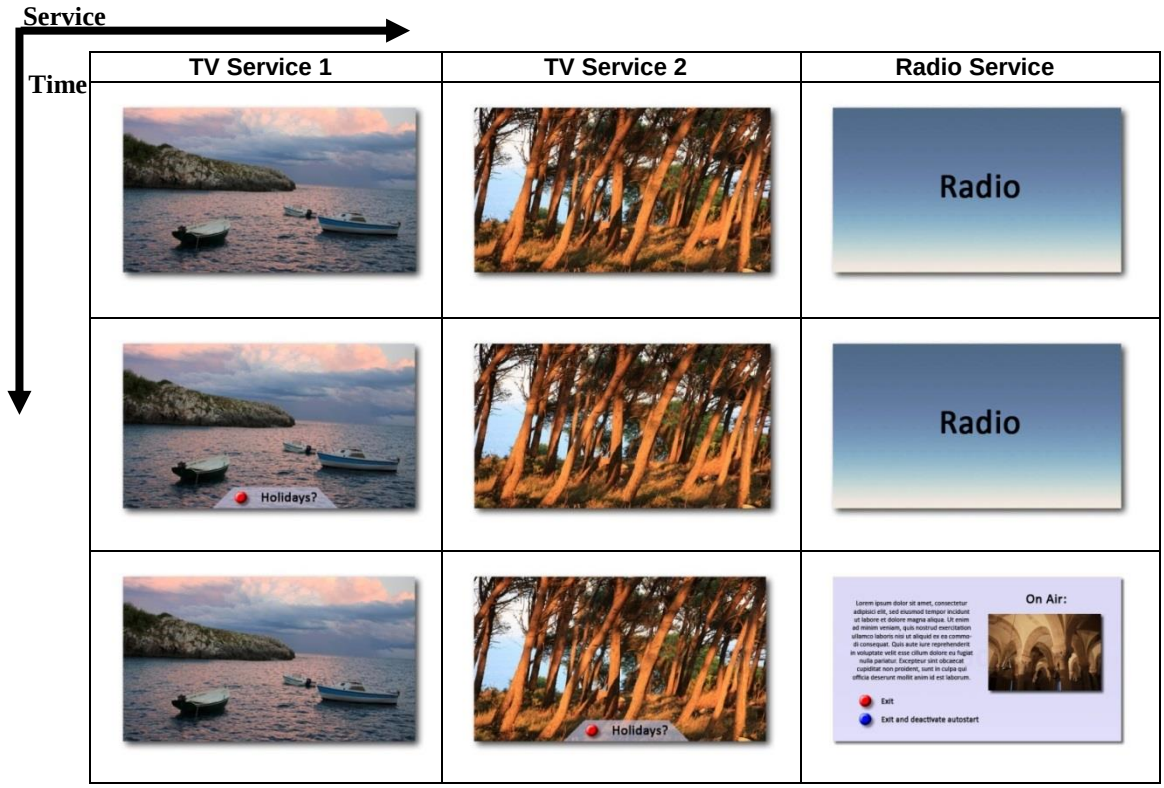


Figure 4: What might be seen across channels and through time

5.2 User input

The user controls interactive applications using a user input device typically supplied with the terminal. This may be a conventional remote control or an alternative input device such as a game controller, touch screen, wand or drastically reduced remote control.

NOTE: While the alternative input devices do not have buttons in the same way as a remote control, it is expected that implementations using these alternative input devices will include means to generate input to the application (called key events) logically equivalent to pressing buttons on a conventional remote control.

Table 2 lists the buttons or key events which are relevant for the end user when using interactive applications. Requirements on implementations are found in Table 12 in clause 10.2.2.

Table 2: Relevant remote control buttons or key events for the end user when using interactive applications

Button or Key Event	Usage
TEXT or TXT or comparable button	Launches the digital teletext application and/or the standard teletext as described in clause 5.3.4.
red colour button	Usually displays or hides a broadcast-related autostart application.
3 additional colour buttons (green, yellow, blue)	Variable usage as defined by the application (typically short-cuts or colour-related functions).
4 arrow buttons (up, down, left, right)	Variable usage as defined by the application (typically focus movement or navigation through lists).
ENTER or OK button	Variable usage as defined by the application (typically selection of focused interaction elements or confirmation of requested actions).
BACK button	Variable usage as defined by the application (typically going back one step in the application flow).
2 program selection buttons (e.g. P+ and P-)	If available: selects the next or previous broadcast service in the internal channel list which may lead to the termination of the running application as described in clause 6. These functions remain active at all times while broadcast-related applications are running – see clause 6.2.2.2.
WEBTV or comparable button	If available: opens a menu providing access to broadcast-independent applications as described in clause 5.3.5.
EXIT or comparable button	Terminates a running broadcast-related application and returns to last selected broadcast service (see clause 6.2.2.3). A running broadcast-independent application is still terminated but what happens next is implementation dependent. For example, it may vary depending on how the terminated application was originally started.

Additionally, users may interact with an HbbTV® application via a Companion Screen application in place, or in addition to, the standard user input device supplied with the terminal. The Companion Screen application does not need to replicate the functions of, and may support features not available on, the standard user input device.

5.3 Access to interactive applications

5.3.1 Overview of ways of access

The end user can access interactive applications via the following ways:

- Accessing a typical broadcast-related autostart application by pressing the visually indicated "Red Button" (see clause 5.3.3.2).
- Starting a digital teletext application by pressing the TEXT button (see clause 5.3.4).
- Starting a broadcast-independent application through the Internet TV portal of the manufacturer if one is offered (see clause 5.3.5).
- Starting an HbbTV® application on the terminal from an already running Companion Screen application.
- Starting an application via a link in the currently running application.
- Selecting a broadcast channel which has a broadcast-related autostart application which starts in full-screen mode (usually only used on radio or data services).

5.3.2 Inaccessibility of applications

If a non-autostart application (e.g. a digital teletext application) is not available via the broadcast channel but only via broadband and the terminal is not connected to a broadband network, the terminal should display a suitable error message encouraging the end user to connect the device to one. Technical error messages (e.g. HTML status code 404) or a black screen should be avoided.

Despite the device having an active broadband connection, failure to access the initial page of an autostart broadband service should not cause any message (error or otherwise) to be displayed on the screen and disturb the TV watching experience.

5.3.3 Starting broadcast-related autostart applications

5.3.3.1 Possible states of an autostart application

Broadcast-related autostart applications are usually associated with a broadcast channel or an event (or part of an event) on that channel. In the first case, they start as soon as the channel is selected. In the second case, they start through an AIT update (usually co-incident with the start of the event).

Broadcast-related autostart applications may be in one of the following states when they start:

3) Displaying a "Red Button" notification to inform the user that the application is available.

4) Displaying no user interface.

5) Displaying their full user interface (usually only used on radio and data services).

In general, autostart applications on TV services should not display their full user interface (i.e. state 3) automatically. Instead, the user is informed of their availability by the "Red Button" icon (i.e. state 1). Further parts of the application should not be started unless the end-user presses the "Red Button".

Applications will start with a window covering the entire display in order that they can position the "Red Button" notification where they wish. Since the browser rendering canvas default colour is device-dependent, applications should explicitly set the background of their `<body>` element to transparent using (for example) the following CSS rule:

```
body {
  background-color: transparent;
}
```

This ensures that the video for the current service is visible in those areas of the screen where the "Red Button" notification is not displayed.

On some services (e.g. radio), a broadcast-related autostart application may start by displaying its full user interface (i.e. state 3) immediately without displaying a "Red Button" icon beforehand.

When an application changes from state 1 or 3 to state 2, it should:

- Remove all graphics on screen.
- Stop presenting any kind of streaming audio or video.
- Restart the broadcast service (if it is a broadcast-related application and the broadcast service has been stopped).
- Rescale/reposition video to "full screen mode" (if video has been scaled/positioned).
- Unmute audio (if audio has been muted).
- Stop consuming any key events apart from the "Red button" (which should be used to change back to state 3).

When an application changes from state 2 to state 1 or 3, it should:

- Show new application graphics as appropriate.
- Inform the terminal which key events it wishes to consume in its new state.

For some use cases e.g. interactive radio applications, some of these may not apply.

5.3.3.2 "Red Button" applications

This type of broadcast-related autostart application indicates its availability by displaying a "Red Button" icon on the screen. This icon is displayed for a time period and then it may disappear. Pressing the "Red Button" on the RCU always displays the full user interface of the application (see figure 5), whether the "Red Button" icon currently being displayed or not. If there is no broadcast-related autostart application, pressing the "Red Button" has no effect (see figure 6).

NOTE: The "Red Button" icon is generated by the broadcast-related autostart application and therefore it is also designed by the owner of the application.

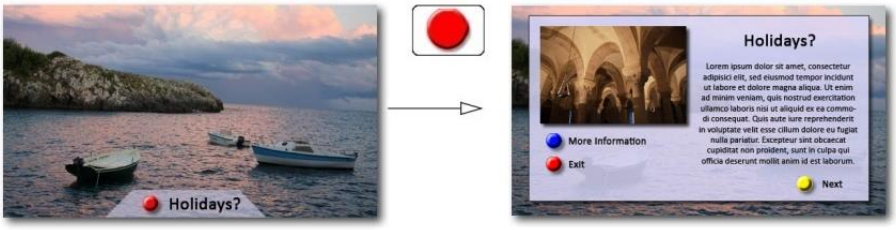


Figure 5: Service with associated broadcast-related autostart application

Figure 6: Service without associated broadcast-related autostart application

The end user may be able to control a setting to disable the initial drawing of the "Red Button" indication. If the end user selects this setting then this broadcast autostart application will display its full user interface when it starts, without drawing a "Red Button" indication. Support for this setting is provided entirely by the application. If such a setting is available, it should be easy for the end user to find and its purpose should be clear to the end user.

5.3.4 Starting digital teletext applications

A digital teletext application is a special broadcast-related application which is started by pressing the TEXT button on the RCU. Depending on the provision of a digital teletext application and of standard teletext the reaction on pressing the TEXT button differs.

Case A: If only the standard teletext is available on the current service, the standard teletext is displayed.

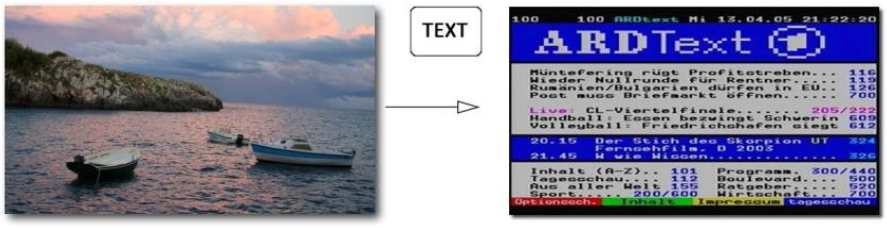


Figure 7: Service with standard teletext only

Case B: If only a digital teletext application is available on the current service, this application is started. Pressing the TEXT button a second time terminates the application and causes the AIT to be re-parsed and any autostart application to be restarted.

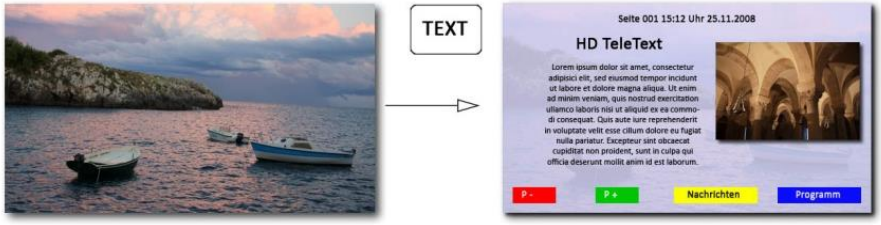


Figure 8: Service with digital teletext application only

Case C: If both a digital teletext application and standard teletext are available on the current service, an easy to use mechanism should be implemented to toggle between the different teletext modes.

EXAMPLE: Pressing the TEXT button for the first time could start the digital teletext application, pressing it for the second time would close the digital teletext application and start the standard teletext, and pressing it for the third time would close the standard teletext and rerun AIT parsing and start the autostart application if provided.

Figure 9: Example of service with digital teletext application & standard teletext

Case D: If a digital teletext application is signalled but not available (because the digital teletext application is only reachable via broadband and the terminal is not connected appropriately) but standard teletext is available, the standard teletext would be displayed (see also figure 7).

Case E: If no digital teletext application is signalled and standard teletext is not available, nothing should happen.

Figure 10: Service without associated teletext

Case F: If a digital teletext application is signalled but not available (because the digital teletext application is only reachable via broadband and the terminal is not connected appropriately) and standard teletext is not available, the terminal would display an informative message encouraging the end user to connect the terminal to the internet.

5.3.5 Starting broadcast-independent applications

Broadcast-independent applications are started via a running application or an Internet TV Portal. An Internet TV Portal is an application which provides a type of start page where broadcast-independent applications are sorted and offered in an appropriate and useful way to the end user. The Internet TV Portal may be opened by pressing a dedicated Internet TV Button on the RCU. The type of interactive applications that are listed in the Internet TV Portal is the responsibility of the manufacturer. There may be an option for the user to add broadcast independent applications via manual URL entry or similar means like apps on mobile phones. The structure and the design of the start page is the responsibility of the manufacturer and out of the scope of the present document. Broadcast-independent applications are described in more detail in clause 6.2.2.6.

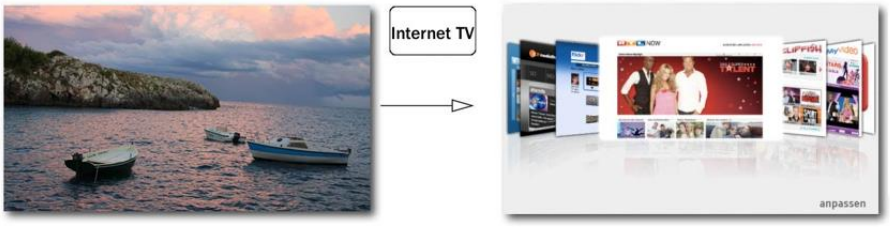


Figure 11: Internet TV Portal

Broadcast-independent applications may also be started from a Companion Screen application running on a Companion Screen. Companion Screen applications and their interaction with HbbTV® terminals and HbbTV® applications are described in more detail in clause 14.

5.4 Exiting and hiding broadcast-related applications

According to the technical definitions of the application lifecycle in clause 6, applications may be stopped when they launch other applications or a channel change is performed. Applications may also kill themselves, either as a result of a request by the end-user or as a consequence of some internal logic.

Pressing the EXIT (or comparable) button terminates the application.

Applications may disappear from view automatically on some actions of the end-user which cause the application to move to state 2 (as defined in clause 5.3.3.1). "Red Button" applications should always provide this function and should use the "Red Button" to toggle between state 2 and state 3 (as defined in clause 5.3.3.1). Applications should use the `Application.hide()` method to hide their user interface, or may use an alternative approach.

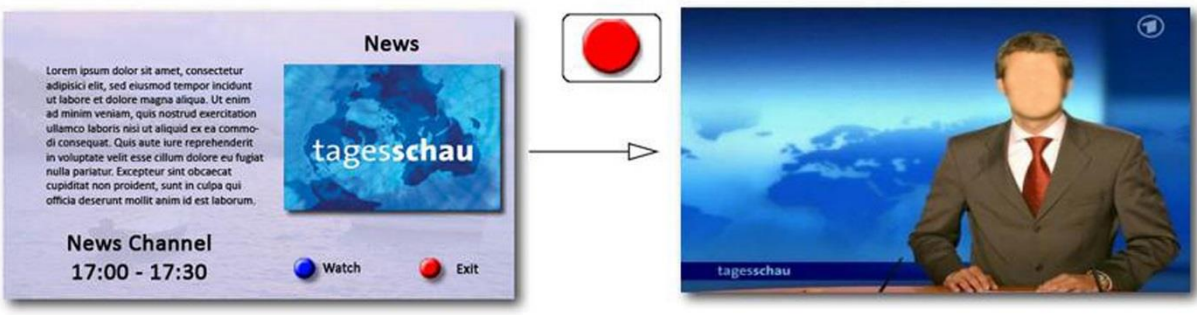


Figure 12: Application selects TV channel

If an action occurs that would terminate an HbbTV® application on the terminal, it is recommended that, where possible, any associated Companion Screen applications are informed of this. This could be achieved using application specific messages via the application to application communication path (see clause 14.5), before the application is terminated.

5.5 Companion Screens

HbbTV® applications may launch, or be launched by Companion Screen applications. Once launched, they may communicate using application specific messages either directly (application to application communication - see clause 14.5) or indirectly (e.g. through the cloud). There may be a multiplicity of Companion Screen applications associated with an HbbTV® application. As a result, there are many application and user interaction models possible and applications should be designed carefully. It can not be assumed by an HbbTV® application that a Companion Screen will be available. It is recommended that the standard user input controls are also be handled by the HbbTV® application (perhaps with reduced application functionality) even when the Companion Screen it intended as the primary input device.

It is also recommended that applications contain assistance and guidance information to help the user to get started with applications or suites of applications that are running on both HbbTV® terminals and Companion Screens.

5.6 User interface issues

5.6.1 Advertising broadcast applications

The user interface displayed on channel change (and when the "Info" button is pressed) is the responsibility of the terminal manufacturer but typically includes the title and synopsis of the current event. It is recommended that the presence of HbbTV® applications signalled in the broadcast is indicated to the user in this UI.

5.6.2 Co-existence with CI and CI Plus MMI

A CICAM may request the terminal to display an MMI screen or dialogue at any time. The terminal has to respect the mandatory requirements of the CI and CI Plus specifications (see clauses 12.3.3 and 12.6.1.1 of CI Plus [12]) and clause 12.4 of the DVB CI Plus Extensions TS 103 205 [37]. Working within those constraints, the terminal should endeavour to present a consistent and uncomplicated user interface at all times. On occasion, this may result in the HbbTV® application at least losing focus and possibly being terminated.

If any interaction between the CICAM and the user is required, application authors are strongly recommended to use the oipfDrmAgent APIs to allow communication between the CICAM and the HbbTV® application, which can then act as a proxy for any interaction with the user.

5.6.3 Encrypted channels

Terminals may wish to display a message to the user that the channel is encrypted and cannot be displayed (see clause 6.2.2.8). If they do so, they should be aware that applications may wish to present some relevant information for this scenario. Hence any native UI should not remain on screen permanently or should give the user a way to remove it.

6 Service and application model

6.1 Application model

The present document defines a model which supports one HbbTV® application visible at one time.

Two types of applications are supported:

- Broadcast-related applications. These are signalled as part of a broadcast channel as defined in clause 7.2.3.1 and follow the lifecycle rules defined in clauses 6.2.2.2 and 6.2.2.3.
- Broadcast-independent applications. These are either not signalled at all or are signalled as in clause 7.2.3.2. They follow the lifecycle rules defined in clause 6.2.2.6.

Applications may transition between these two types as described later in the present document.

Terminal specific applications like navigators, channel list management, terminal specific EPGs or PVR control applications are out of scope of the present document.

No mechanism is defined to allow the visible application to interact with other running applications.

Terminal specific applications may be temporarily displayed on top of HbbTV® applications. This shall not affect the state of the HbbTV® application but during this time, if the terminal specific application takes focus, the HbbTV® application shall not receive any key event. Calls to `application.show()` while a terminal specific application is visible shall either:

- cause the HbbTV® application to be visible behind the terminal specific application; or
- cause the HbbTV® application to become visible once the terminal specific application stops being visible assuming that the HbbTV® application is still running and that `application.hide()` has not been called.

6.2 Application lifecycle

6.2.1 Introduction

The application lifecycle is determined by the following four factors:

1) The application model.

2) The currently selected broadcast service (if any) and changes to it.

3) The applications signalled as part of the currently selected broadcast service.

4) The signalled application control code (as defined in clause 7.2.3.1 of the present document and clause 5.2.4 of TS 102 809 [3]).

6.2.2 Starting and stopping applications

6.2.2.1 Summary (Informative)

Starting an application may be initiated in the following ways:

- Directly by the end-user (e.g. by using dedicated buttons on the remote control or an equivalent menu provided by the terminal).
- In response to signalling in a broadcast service (e.g. automatically starting a broadcast-related autostart application).
- By an already running application (via the JavaScript method `createApplication()`).
- By a Companion Screen as described in clause 14.6.

Starting applications in response to the playback of recorded or downloaded content is not supported.

An application may be stopped in the following ways:

- As defined in the flowcharts in clauses 6.2.2.2 and 6.2.2.3.
- By calling `Application.destroyApplication()`.
- By the terminal, under certain error conditions as defined in clause 6.2.2.11.
- Directly by the end-user.

The launch of an application may be blocked by the terminal if a parental rating value is signalled in the broadcast AIT or XML AIT. However, once launched, a change in the parental rating in a broadcast AIT does not cause the running application to be killed.

6.2.2.2 Behaviour when selecting a broadcast service

Figure 13 shows the rules that shall apply when the selected broadcast service changes.

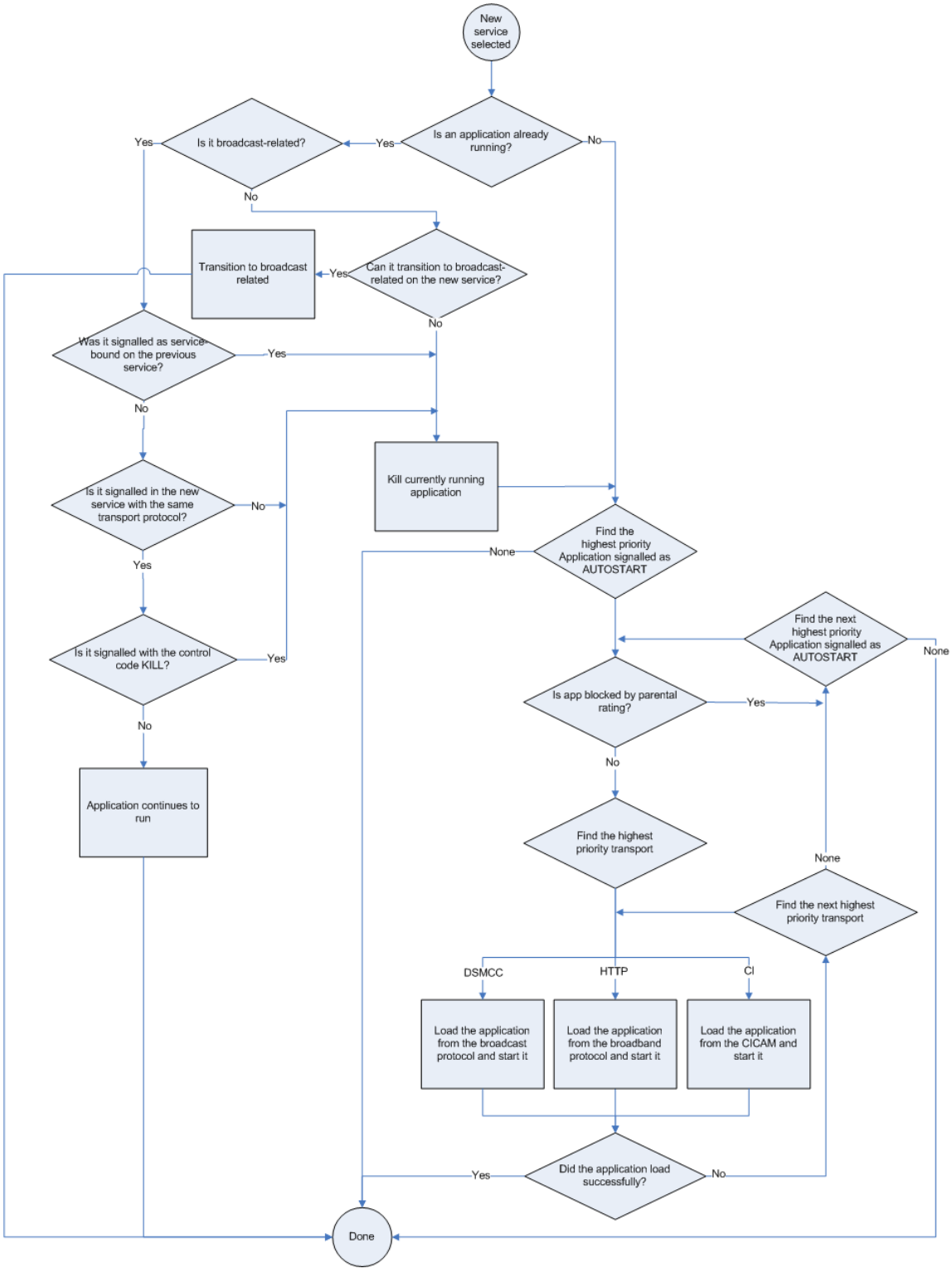


Figure 13: Behaviour when selecting a broadcast service

NOTE 1: It is strongly recommended that broadcasters only signal one autostart application per broadcast service.

NOTE 2: The selection of application can be optimized as follows:

If the terminal does not have an operational broadband connection then applications signalled as broadband-only and broadband-specific signalling for applications signalled as both broadcast and broadband can be discarded.

If the terminal does not have an operational CI Plus connection then applications signalled as CICAM-only and CICAM-specific signalling for applications signalled as both broadcast and CICAM-based can be discarded.

The channel change mechanisms offered by the terminal (e.g. P+/P- keys, number keys) shall remain functional at all times while broadcast related applications are running, regardless of whether media is being presented and whether that originates from broadcast or broadband.

For the purposes of deciding whether an application is already running or is signalled, only the `organisation_id` and `application_id` fields from the AIT shall be used. Other information (e.g. the URL of the first page, the parental rating of the application) shall not be used.

No application shall be launched by the terminal if it would be blocked by parental access controls, as defined in clause 6.2.2.10.

Figure 13 shall not apply when selecting an MPEG program which is not a broadcast DVB service. If a transport stream does not include an SDT actual then none of the MPEG programs in that stream are broadcast DVB services. If the SDT actual in a transport stream does not include an entry corresponding to a PMT in that transport stream then the MPEG program described by that PMT is not a broadcast DVB service. There is no requirement for a terminal to check again either for an SDT or that a service is listed in the SDT if it has already done so, e.g. in order to acquire the service name when creating the channel list.

NOTE 3: If broadcasters or operators change programs in a multiplex from being a broadcast service to a non-broadcast service or vice-versa, they should use new program numbers/service_ids and should not re-use the old program numbers/service_ids.

As a consequence of selecting such an MPEG program:

- No applications shall be started.
- No applications shall be stopped except for broadcast-related applications with `service_bound_flag` set to '1' which are stopped when leaving the current broadcast service.
- The value of the `currentChannel` property on the video/broadcast object and the `ApplicationPrivateData.currentChannel` property shall reflect the MPEG program selected.
- Figure 13 shall not apply when selecting an MPEG program that is not a broadcast DVB service.

6.2.2.3 Behaviour while a broadcast service is selected

Figure 14 shows the rules that shall apply if the AIT changes or a broadcast-related application exits while a broadcast service is selected.

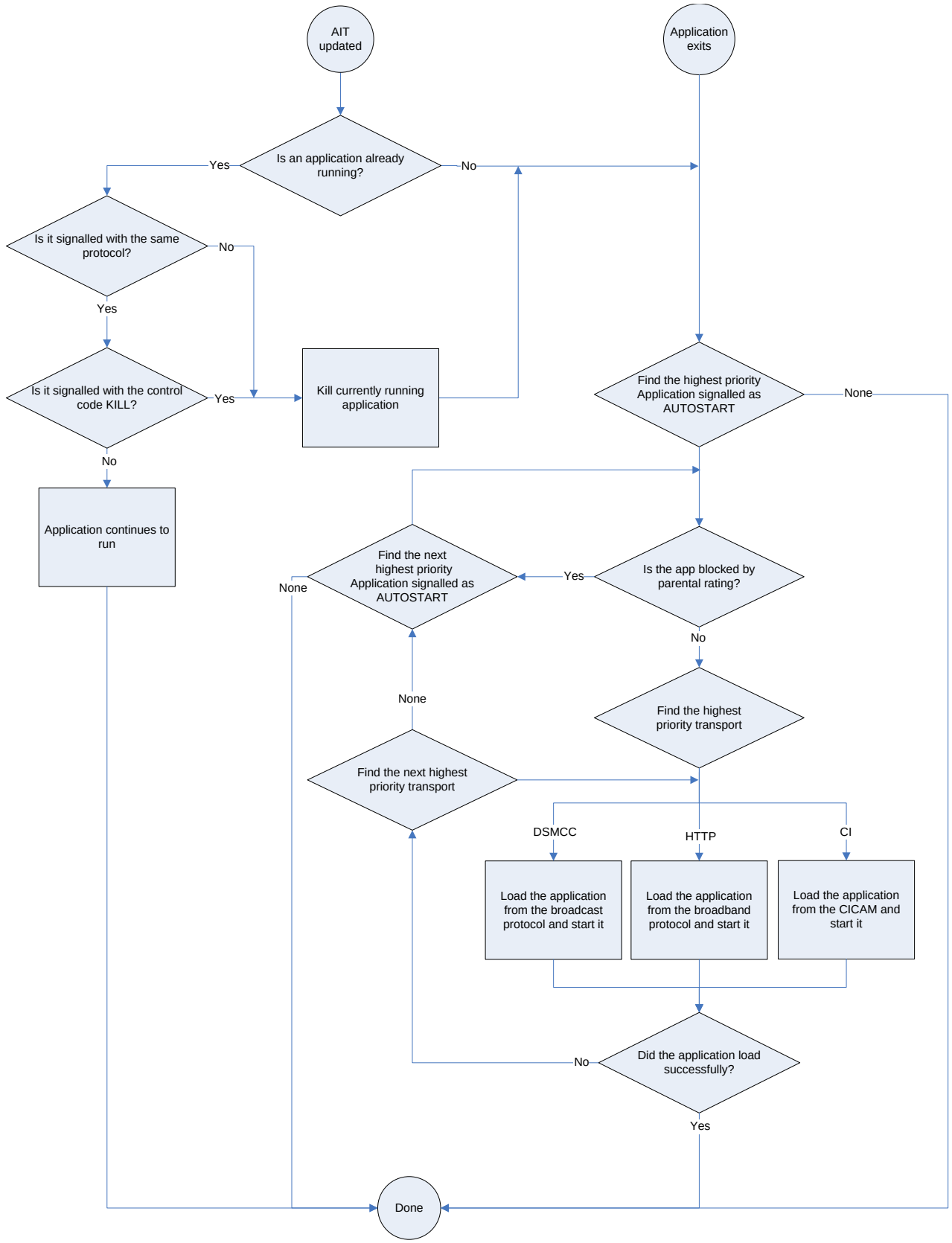


Figure 14: Behaviour while a broadcast service is selected

No application shall be launched by the terminal if it would be blocked by parental access controls, as defined in clause 6.2.2.10.

In figure 14, the following clarifications shall apply:

- For the purposes of deciding whether an application is already running or is still signalled, only the `organisation_id` and `application_id` fields from the AIT shall be used. Other information (e.g. the URL of the first page, the parental rating of the application) shall not be used.

- Other than `organisation_id` and `application_id`, the only other field in the AIT which is relevant when the AIT is updated is the application control code. Changes in other fields shall be ignored for already running applications.

NOTE 1: As a result of the above, changes to fields in the AIT other than `organisation_id`, `application_id` and application control code will only take effect for newly started applications. In order for those changes to affect an already running application, the application needs to exit and re-start. It is up to the broadcaster and/or application provider to arrange for this to happen.

NOTE 2: A change in the version number of an AIT subtable is an indication to the terminal to retrieve a new version of the AIT. It does not imply or require any changes in the content of the AIT itself. For example, adding an application to the AIT would be an update to the AIT without changing the AIT entries for any existing applications.

NOTE 3: The selection of application can be optimized as follows:

If the terminal does not have an operational broadband connection then applications signalled as broadband-only and broadband-specific signalling for applications signalled as both broadcast and broadband can be discarded.

If the terminal does not have an operational CI Plus connection then applications signalled as CICAM-only and CICAM-specific signalling for applications signalled as both broadcast and CICAM-based can be discarded.

The PID on which an AIT component is carried may change. Terminals shall treat this in the same manner defined in clause 5.3.4.2 of TS 102 809 [3] for the case where an AIT is removed from the PMT and then reinstated. This means that the subtable shall be considered to have changed, regardless of whether the AIT version number changes, and the normal "AIT updated" sequence defined in figure 14 shall be followed.

If the only running broadcast-related application exits without starting a broadcast-independent application or without the terminal changing channel, the AIT shall be re-parsed and any autostart application shall be re-started following the rules defined in the previous clause. It may be that the restarted application is the same one as the one that just exited. If an application exits when an MPEG program that is not a broadcast DVB service is selected and that MPEG program does not include an AIT then the behaviour is implementation specific.

This flowchart shall not apply while MPEG programs are selected which are not a broadcast service, (i.e. not listed in the SDT of the transport stream carrying them or are carried in a transport stream that does not include an SDT) and which do not include an AIT.

Terminals shall include a mechanism to start and stop digital teletext applications, for example, the TEXT key on an RCU could be used to start the digital teletext application (which would require any other running application to be killed); pressing the TEXT key again causes the running application to be stopped as long as it is signalled as a digital teletext application. Digital teletext applications are identified with an `application_usage_descriptor` in the AIT with `usage_type` equal to `1`.

NOTE 4: The digital teletext application is intended to be distinct from the autostart application(s) in the AIT. Care is needed if a teletext application is started by means other than the TEXT key.

6.2.2.4 Time-shifting behaviour

If the terminal initiates time-shifting of the currently selected broadcast service, an application may get out of sync with the presentation of the audio-video components of this service. An HbbTV® application shall be terminated if it is not safe to run it on a time-shifted broadcast service. An application is safe to run in time shift mode, if it is signalled in the AIT with an `application_recording_descriptor` and both the `trick_mode_aware_flag` and the `time_shift_flag` set to '1' as described in clause 7.2.3.1. If an application is killed due to a broadcast service being time-shifted, the procedure defined in clause 6.2.2.2 for selecting an autostart application to run shall be followed except that only applications that are time-shift safe shall be considered.

After starting time-shift a terminal shall:

- Dispatch a `onPlaySpeedChanged` event with a speed other than 1.0 to signal that time-shift has started.
- Update the `currentTimeShiftMode`, `playPosition` and `playSpeed` properties of the video/broadcast object.

The present document defines two implementation options for support of applications when video is time-shifted - depending on whether the terminal can or cannot maintain synchronization between applications and the A/V components of a service. Which of these two options is implemented by a terminal is indicated by the `timeShiftSynchronized` property.

When a terminal can maintain synchronization between applications and the A/V components of a service, all of the following shall apply:

- DSM-CC stream event descriptors shall be recorded with the A/V components keeping the timing relation and shall be delivered during playback of the time-shift.
- The AIT shall be monitored, any changes shall take effect preserving the correct timing with respect to the A/V components.
- The service information shall be recorded with the A/V components keeping the timing relation and the properties of the video broadcast object (e.g. programmes, `AVComponent` as defined in clause 7.16.5 of the OIPF DAE specification [1]) changes at the proper time of the playback of the time-shift.
- The `timeShiftSynchronized` property of the `configuration` property of the `Configuration` class shall be set to true (see clause A.2.20.2).

If a terminal is not able to maintain synchronization between applications and the A/V components of a service:

- The application may receive some (or all) broadcast resources from the live broadcast signal instead of the time shift playback.
- It shall set the `timeShiftSynchronized` property to false.

NOTE: When an application accesses service information or receives stream events, it may check if it is synchronized with the A/V component of the service by reading the values of the properties `recordingState` and `timeShiftSynchronized`.

6.2.2.5 Simultaneous broadcast/broadband/CI Plus application signalling

6.2.2.5.1 Priority

Broadcast, broadband and CI Plus transport protocols may be specified simultaneously for a given application. The priority by which the transport protocols shall be used is determined by the order in which the `transport_protocol_label`s are listed in the `application_descriptor`, with the first being the highest priority.

6.2.2.5.2 Not currently operational broadband connection

Where a terminal does not have a currently operational broadband connection and an application to be launched is signalled to be:

- Available through broadband and one or more other protocols (either broadcast or CI Plus): the terminal shall disregard the signalling for the broadband transport protocol.

- Available only through broadband: the terminal shall ignore the request to launch the application (and return an error if the application was launched by a call to `createApplication()`).

6.2.2.5.3 Currently operational broadband connection and error accessing initial page

Where a terminal has a currently operational broadband connection but there is an error (asynchronous due to the nature of the HTTP protocol) accessing the initial page of a broadband application and an application to be launched is signalled as:

- Available through broadband as top priority and then through another protocol (either broadcast or CI Plus): the terminal shall disregard the signalling for the broadband transport protocol.

- Available only through broadband: the terminal shall not display an error message for applications which were either launched as autostart (e.g. following a channel selection or AIT update) or which were launched by another application.

6.2.2.5.4 Not currently operational CI Plus protocol

Where a terminal does not have a currently operational CI Plus File System and an application to be launched is signalled to be:

- Available through CI Plus protocol and one or more other protocols (either broadcast or broadband): the terminal shall disregard the signalling for the CI Plus transport protocol.

- Available only through CI Plus protocol: the terminal shall ignore the request to launch the application (and return an error if the application was launched by a call to `createApplication()`).

6.2.2.5.5 Currently operational CI Plus connection and error accessing file system

Where a terminal has a currently operational CI Plus File System with HbbTV® Application Domain but there is an error accessing the initial page of a CI Plus application and an application to be launched is signalled as:

- Available through CI Plus protocol and one or more other protocols (either broadcast or broadband): the terminal shall disregard the signalling for the CI Plus transport protocol.

- Available only through CI Plus protocol: the terminal shall ignore the request to launch the application (and return an error if the application was launched by a call to `createApplication()`).

6.2.2.5.6 Application launch failure

If the application cannot ultimately be loaded from either broadcast or broadband or CI Plus and the application was launched by a call to `createApplication()`, an `ApplicationLoadError` shall be dispatched. Once the initial page of an application has been successfully loaded, the present document does not specify how terminals should behave if a page from that application subsequently fails to load.

6.2.2.6 Broadcast-independent applications

6.2.2.6.1 Lifecycle issues

A broadcast-independent application can be created in one of the following ways:

- By calling the `Application.createApplication()` method with either an HTTP or an HTTPS URL. The URL shall refer to either an HTML page or an XML AIT (see clause 7.2.3.2).

- Optionally from a terminal specific application like an Internet TV Portal or following manual URL input as described in clause 5.3.5. See also clause 6.2.2.6.2.

- By a Companion Screen sending an XML AIT to the terminal as described in clause 14.6.

Where the URL refers to an HTML page directly, the broadcast-independent application shall be created without an `organisation_id` or `application_id`.

Where the URL refers to an XML AIT, the broadcast-independent application shall be created with the `organisation_id` and `application_id` specified in the XML AIT. In both cases, the application shall be associated with an application boundary as defined in clause 6.3.

When a broadcast-related application starts a broadcast-independent application, the application is started but the broadcast service shall cease to be selected - logically equivalent to selecting a "null service" as described above. The new application shall not have access to broadcast resources. When a broadcast service ceases to be selected, the terminal shall stop the presentation of any component of that broadcast service. When a broadcast-independent application is running, the terminal shall not present components of a broadcast service.

A broadcast-related application can transition to a broadcast-independent application by calling the `setChannel()` method on the video/broadcast object with a value of `null` for its `channel` argument. Access to broadcast resources shall be lost and the object shall transition to the `unrealized` state. A `ChannelChangeSucceededEvent` shall be dispatched to the video/broadcast object that caused the transition with a value of `null` for the `channel` property.

NOTE 1: Applications that wish to become broadcast-independent and later transition back to broadcast-related should remember the current channel before transitioning to broadcast-independent.

NOTE 2: Broadcast-related applications should ensure that the current HTML page is loaded from broadband before making such a transition. As defined in clause 6.3.3 of the present document, broadcast-related applications loaded from carousel can have their application boundary extended to include HTTP or HTTPS domains in order to enable loading of pages from broadband as part of the application. The results of attempting to make a transition to broadcast-independent when the current HTML page is loaded from carousel are not defined by the present document but may include the application being unable to load any data or it being terminated.

Stopping and resuming playback of broadcast video using the `stop()` and `bindToCurrentChannel()` methods on a video/broadcast object shall not affect the broadcast-related status of an application.

When a broadcast-independent application successfully selects a broadcast service using a video/broadcast object, that application shall be killed unless all the following conditions are met:

- The broadcast-independent application has an `organisation_id` and `application_id` (whether obtained through a broadcast AIT or an XML AIT).
- An application of the same `organisation_id` and `application_id` is signalled in the broadcast channel to be selected with control code `AUTOSTART` or `PRESENT`.
- The application signalled in the broadcast channel with the same `organisation_id` and `application_id` includes a `transport_protocol_descriptor` with `protocol_id` equal to 3.
- The URL of the entry point document of the broadcast-independent application has the same origin as at least one of the URLs signalled in the broadcast for that `organisation_id` and `application_id`.
- The URL of the page currently loaded in the broadcast-independent application is inside the application boundary of the application as defined in clause 6.3.

If these conditions are met, the application shall transition to be a broadcast-related application as defined in clause 6.2.2.2 and hence the broadcast signalling in the successfully selected service shall be obeyed thereafter. The application should be authored to follow the behaviour defined in clause 5.3.3. These conditions shall apply regardless of whether an application was originally launched as broadcast-related or as broadcast-independent and regardless of how many times an application may have previously transitioned from broadcast-related to broadcast-independent or vice-versa.

6.2.2.6.2 Launch context signalling (informative)

As defined in clause 6.2.2.6.1, broadcast-independent applications may be launched from terminal-specific applications. These could include Internet TV portals, programme guides, related content panels, search results, channel banners, voice activation etc.

Where many launch routes are possible for the same application, the application may need to understand the context from which it was invoked if it is to provide the best user experience. If this information is available, the application can show the most relevant information straight away without unnecessary user interaction and offer different onward journeys within the application. This information is also useful to the application provider in understanding how applications and content are discovered through the terminal, thus leading to better user-experiences.

This clause defines a convention for how launch context information can be included in the application URL when starting a broadcast-independent application from a terminal-specific application.

Where this convention is adopted, when the terminal launches a broadcast-independent application, the application URL (which may refer to either an HTML page or an XML AIT) is modified to add a launch context query parameter of the form "`lloc=<launch location>`". This string is added before the first number sign (#) character in the URL if there is one, or at the end if there is not, using either a "?" or a "&" character in order to maintain a legal URL structure as defined in RFC3986 [27].

The following table defines a set of values for "`<launch location>`" and their meanings. Other local or platform specific values may also be defined.

Table 2a: Defined launch location terms

User interface view or terminal-specific application	<launch location> value
Any part of the terminal's primary home screen or portal page.	homepage
Any platform-specific section of the terminal user interface that lists available applications that present A/V media content.	playerpage
Any full-screen view within the terminal's electronic programme guide that shows content available on linear channels.	epg
Any partial-screen view within the terminal's electronic programme guide such as a channel selection banner or 'miniguide'.	miniguide
Any view showing results from a user-initiated content search.	search
Any view showing content recommendations.	recommendations
Any view showing browsable content choices and which does not fall within the categories defined above.	browse
Direct speech command which does not fall within the categories defined above.	speech
Any view that does not fall within the categories defined above and for which no platform-specific or local term is defined.	other

Content providers may use the same convention themselves when launching HbbTV applications from other applications and from companion screen devices. In those cases, the complete application URL is provided directly by the content provider and the content provider may freely choose the string to be used for "<launch location>".

Examples:

1) A broadcast-independent application is available with the URL `http://www.example.com/hbbtv-application`. When it is launched from the terminal's list of available applications, the terminal appends the string "`?lloc=playerpage`" and starts the application as `http://www.example.com/hbbtv-application?lloc=playerpage`.

2) The same application is launched from a companion screen application. The content provider requests the application be launched using the complete URL `http://www.example.com/hbbtv-application?lloc=companion` in order to differentiate from the terminal launch cases.

3) A broadcast-independent application supports "deep linking" to content. The URL for the application that is associated with a particular content item is `http://www.example.com/deeplink?cid=is38g7bv`. When the application is launched from the terminal's electronic programme guide, the terminal starts the application as `http://www.example.com/deeplink?cid=is38g7bv&lloc=epg`. If the same deep link is discovered by a user initiated content search, the terminal starts the application as `http://www.example.com/deeplink?cid=is38g7bv&lloc=search`.

4) A broadcast-independent application is available with the URL `http://www.example.com/hbbtv-application#mode4`. When it is launched from the terminal's list of available applications, the terminal inserts the string "`?lloc=playerpage`" and starts the application as `http://www.example.com/hbbtv-application?lloc=playerpage#mode4`.

6.2.2.7 Access to broadcast resources while presenting broadband-delivered A/V

Terminals shall be able to present broadband delivered video at the same time as demultiplexing MPEG-2 sections from the broadcast. In particular, the following examples shall apply:

- AIT monitoring shall continue.
- Files in a carousel shall be accessible including updates.
- DSM-CC stream event monitoring shall continue.
- `ProgrammesChanged` events shall be sent to any registered listeners.

Broadcast-related applications which wish to present long items of broadband delivered video should either:

a) make themselves broadcast-independent as defined in clause 6.2.2.6; or

b) be permanently signalled in the AIT by the broadcaster.

6.2.2.8 Behaviour on encrypted broadcast services

Some channels may have the broadcast content encrypted, preventing those terminals without the appropriate CAS and rights from decoding and presenting the content. In these cases, clauses 6.2.2.2 and 6.2.2.3 remain applicable even when the terminal fails to decode some or all of the components.

In particular, terminals shall behave as follows:

- Failure to decrypt the AIT is identical to having no AIT present on that channel.
- Failure to decrypt the carousel containing the application is identical to failing to load the application from broadcast protocol.

NOTE: The present document is intentionally silent about requirements for terminals to support decryption of encrypted AITs, object carousels and other data components.

Applications associated with channels which may be encrypted are advised to check whether the content is being presented (using the `error` parameter provided in the `onPlayStateChange` method of the video/broadcast object) and to modify their behaviour accordingly. For instance, if the content is not being presented, the application may wish to display some advertising message indicating how the user may gain access to this channel. Applications should not remain hidden or show a mainly transparent screen.

6.2.2.9 Applications launched from non-HbbTV® application environments

Terminals may support broadcast application types other than HbbTV® applications and may provide a mechanism for an HbbTV® application to be launched from those application types. The following clauses apply where an HbbTV® application is launched using an XML AIT from an application that is signalled in a broadcast service but which is not an HbbTV® application.

Where a broadcast AIT that includes the HbbTV® `application_type` is present in the service from which the launching (non-HbbTV®) application was running, and if all of the conditions for an application to survive a transition from broadcast-independent to broadcast-related in clause 6.2.2.6 apply then the HbbTV® application shall be started as a broadcast related application and hence the broadcast signalling shall be obeyed thereafter. Otherwise, the application shall be started as broadcast independent.

Where either no AIT is present in the service from which the launching (non-HbbTV®) application was running, or a broadcast AIT is present in that service but does not include the HbbTV® `application_type`, the HbbTV® application shall be started as a broadcast related, non service-bound application. In this case, `ApplicationPrivateData.currentChannel` shall be set to reflect the current channel at the time of the HbbTV® application launch.

In all cases, the HbbTV® application is subject to the normal application lifecycle behaviour on any subsequent channel changes and updates to the broadcast AIT (including its appearance or disappearance).

6.2.2.10 Parental ratings

When an attempt is made to launch an application using any of the mechanisms defined in clause 6.2.2, the terminal shall enforce parental access control on the application being launched using any parental rating information carried in

the AIT. The decision making process and any UI should be the same as that which the terminal would use to enforce parental access control when attempting to play a media content item.

NOTE: Whether the terminal enforces parental access control when launching applications may depend on the configuration of the terminal and in particular parental control settings that have been configured by the user.

If playback of a media content item with the same parental rating would be blocked (e.g. because the user does not enter the correct parental control PIN, or because the terminal is configured to automatically block consumption of content above a given parental rating or if none of the parental ratings provided in the broadcast AIT or XML AIT are supported by the terminal), the request to launch the application shall fail.

6.2.2.11 Other general behaviour

Any application shall be stopped under the following circumstances:

- The application itself exits using the `Application.destroyApplication()` method (as defined in clause 7.2.2 of the OIPF DAE specification [1]).
- In response to changes in the application signalling as defined in clauses 6.2.2.2 and 6.2.2.3 for broadcast-related applications.
- The terminal has run out of resources for executing the application (except as described below) and therefore has to terminate it in order to keep operating correctly.
- The user manually terminates the application using the "EXIT or comparable button" mechanism as defined in clause 10.2.2.1.

An application shall not be stopped due to a failure to load an asset (e.g. an image file) or a CSS file due to a lack of memory, although this may result in visual artefacts (e.g. images not being displayed). Failure to load an HTML or JavaScript file due to a lack of memory may cause the application to be terminated.

See clause 6.2.2.3 for the definition of the behaviour that shall apply when a “broadcast-related application exits without starting a broadcast-independent application or without the terminal changing channel”.

By default, newly started broadcast-related applications shall not be visible to the end user. These applications shall call the `Application.show()` method in order to display their user interface and accept user input. Newly started broadcast-independent applications shall be visible and active without needing to call this method.

Terminals may be configurable (either by the user or by the manufacturer) to not load or not start applications in spite of other requirements in the present document.

The requirements in the present document on starting and stopping HbbTV® applications may be modified for markets where other application formats are already deployed. For example, a static priority (one format always having priority over another where both are present) or a dynamic priority based on broadcast signalling may be used.

When one application requests a second application be started, the first application shall continue to run until the initial HTML document of the second application has been loaded - i.e. until after an `ApplicationLoadError` event would be generated (if any listener was registered). Only then shall the first application be stopped by the terminal.

Failing to parse the initial page of an application shall be regarded as a loading failure when evaluating if the application successfully loads in figures 13 and 14.

When an application selects a new broadcast channel, there is a period of time between the channel change having been completed (when the `onChannelChangeSucceeded` event is triggered) and the AIT having been received and parsed. During this period, the application shall retain its type (broadcast-related or broadcast-independent) and trust level (trusted or untrusted). Hence, while a broadcast-independent application is transitioning to become broadcast-related, access to features limited to broadcast-related applications will continue to fail as they did before the transition started until the AIT has been received and parsed.

6.2.3 Application lifecycle example (informative)

Figure 15 and Table 3 illustrate the application model defined above.

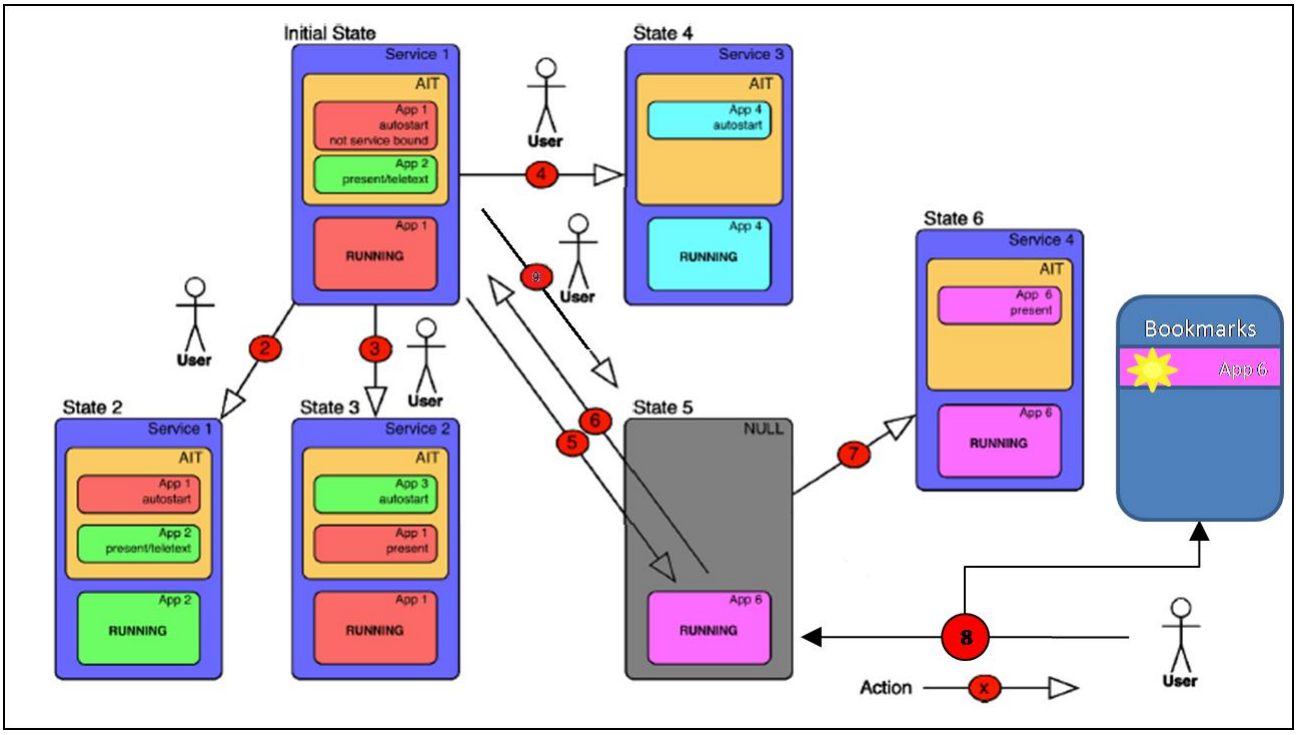


Figure 15: Application model examples

Table 3: Descriptions of actions and resulting state changes

Starting state	Action	Resulting state
Initial State: Application 1 is running	2: User presses "TEXT" key	State 2: Application 2 will be started due to `TELETEXT` signalling.
Initial State: Application 1 is running	3: User selects service 2	State 3: Application 1 keeps running assuming it is not service-bound.
Initial State: Application 1 is running	4: User selects service 3	State 4: Application 1 will be killed and Application 4 will be started due to `AUTOSTART` signalling.
Initial State: Application 1 is running	5: Application call to `createApplication()` with an XML AIT to start a broadcast-independent application	State 5: Broadcast-independent application 6 is running. Any former presentation of service components will be stopped. The application has an application identifier as it was started from an XML AIT. See also action #7.
State 5: Application 6 is running	6: User selects Service 1	State 1: Application 6 will be stopped and Application 1 will be started due to `AUTOSTART` signalling.
State 5: Application 6 is running	7: Application 6 selects service 4	State 6: Presentation of service 4 starts. Application 6 is signalled on service 4. It transitions to broadcast-related and keeps running.
	8: User enters URL of XML AIT or initial page to start application and to store it in his bookmarks. Terminal takes application title and logo for bookmark entry as signalled in HTML header.	State 5: same as for action 5.
	9: Companion Screen sends an XML AIT to the terminal.	State 5: same as for action 5.

6.3 Application boundary

6.3.1 Introduction

Every application is associated with an application boundary. The application boundary is based on origins of the application resources like HTML documents.

6.3.2 Origin

- For resources loaded via HTTP and HTTPS, the origin shall be as defined in clause 5.3 of the HTML5 Recommendation [54].
- For resources loaded via DSM-CC object carousel, the origin shall be the DVB URI in the form (as defined in TS 102 851 [10] clause 6.3.1):
  - "dvb" ":" "//" original_network_id "." transport_stream_id "." service_id "." component_tag.

NOTE: In this case, the "host" is the DVB triplet plus the component_tag.

Hexadecimal digits in the DVB triplet and the component_tag shall be encoded using lower case characters.

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 42 and with Web Storage.

6.3.3 Application boundary definition

The application boundary is defined as follows:

- An application boundary is a set of origins where each origin is as defined above.
- If the origin of a URL is the same as one of the origins in the application boundary, that URL is said to be inside the application boundary.
- For applications loaded via DSM-CC, the default application boundary shall include the origin, as defined in clause 6.3.1, of the initial HTML document used to launch the application.
- If an object carousel is identical to one of the carousels in the application boundary, that carousel is said to be inside the application boundary:
  - The requirements for two object carousels to be identical shall be as defined in clause B.2.10 of TS 102 809 [3].

NOTE 1: For carousels delivered by different transport streams, the terminal compares the two `carousel_ids`. The use of the broadcaster's `organisation_id` in the 24 MSBs of the two `carousel_ids` is a means to obtain unique `carousel_ids` and is not visible to the terminal.

- For applications loaded via HTTP or HTTPS, the default application boundary shall include the origin of the URL used to launch the application e.g. as signalled in the AIT or XML AIT or passed as argument of `createApplication()`.

NOTE 2: This means that the default boundary is the tuple (scheme, host, port) of the application URL before any redirect, where the port component is the default port if not otherwise specified.

- A `simple_application_boundary_descriptor` may be present in the AIT or an `<applicationBoundary>` element may be present in the XML AIT. As described in clauses 7.2.3.1 and 7.2.3.2 of the present document, these may include:
  - one or more `http:` or `https:` URLs prefixes. The application boundary shall be extended to include also the origins of such prefix if this will not result in having origins from more than one host in the boundary. Otherwise the additional origin shall be ignored.

NOTE 3: This means that the boundary cannot be extended to cover more than one FQDN.

  - one or more `dvb:` URL prefixes. The application boundary shall be extended to include also object carousels referenced by such prefixes.

- For applications loaded from the CICAM Auxiliary File System, the application boundary shall include the origin of the URL used to launch the application e.g. as signalled in the AIT or XML AIT or passed as argument of `createApplication()`.

- Extensions to the application boundary shall have no effect on the origin that is used for the enforcement of the same-origin security policy.

Launching a new application by using the method `createApplication()` (with an arbitrary new start page) or killing the current application and starting a new one via application signalling shall result in losing the association with the current application boundary (i.e. the new application will have a new boundary as defined in this clause).

Documents loaded from outside the application boundary shall be untrusted (in the sense of the word "trusted" as defined in clause 11), for example documents loaded in an `<iframe>` element or documents loaded as a result of following a link or an HTTP redirect. Following a link or an HTTP redirect from outside the application boundary back inside the application boundary shall restore the trust level to the original trust level of the application.

NOTE 4: An application being broadcast-related or broadcast-independent is not impacted by this change in trust level.

7 Formats and protocols

7.1 General formats and protocols

7.1.1 Graphic formats

The graphics formats used shall comply with clause 9.1 of the OIPF media formats specification [2].

Table 4 lists the graphics formats that shall be supported.

Table 4: Graphics formats

Image Format	MIME Type
JPEG	image/jpeg
GIF	image/gif
PNG	image/png

7.1.2 Audio description

For the broadcast connection, signalling of audio description is defined by the appropriate specifications for each market where the terminals are to be deployed. Signalling of audio description for MPEG-2 transport streams delivered by the broadband connection shall follow the specification for the broadcast connection (if any).

NOTE: Typically most countries will use one of the 3 mechanisms from clause 8.4.2 of the OIPF DAE specification [1] but the present document does not require that.

For ISO format files, signalling is only defined to identify audio description streams when these are delivered using DASH. In this case, the signalling is defined in clause 6.1.2 of the DVB DASH profile TS 103 285 [45], "Role Related Requirements".

Presenting a broadcast-mix audio description stream is supported since this is no different from presenting any other alternative audio stream.

Presenting receiver-mix audio description streams is not required by the present document.

To the extent that audio description is supported, it shall be exposed to applications as defined in clause 8.4.5 of the OIPF DAE specification [1].

7.2 Broadcast-specific format and protocols

7.2.1 System, video, audio and subtitle formats

The present document does not contain any requirements for system, video, audio and subtitle formats for the broadcast channel. These requirements are defined by the appropriate specifications for each market where the terminals are to be deployed.

7.2.2 Protocol for application transport

DSM-CC object carousel as defined in clause 7 of TS 102 809 [3] shall be supported.

Broadcasters shall ensure that the DSM-CC sections for a carousel are distributed over 3 or fewer elementary streams. StreamEvent sections may be carried in additional elementary stream(s).

Support for the `caching_priority_descriptor` as defined in clause B.2.2.4.2 of TS 102 809 [3] is not included. Clause B.5.2 of TS 102 809 [3] specifies that transparent caching is the default caching level in the absence of this descriptor.

The use of the `deferred_association_tags_descriptor` for the purpose of referencing an elementary stream (TS 102 809 [3], clauses B.3.1.1 and B.3.2) is not required by the present document. However this signalling may be present in a broadcast transport stream and acted upon by receivers that support this. Consequently, authors/broadcasters/operators should not expect this signalling to be ignored if it is present in the broadcast transport stream.

If elementary streams present in other services are to be referenced, then that elementary stream will also be required to be present in the current service's PMT.

The use of the `deferred_association_tags_descriptor` to support the `BIOP_PROGRAM_USE` tap (TS 102 809 [3], clause B.3.1.2) is required by the present document.

The elementary streams used to carry DSM-CC object carousel sections may additionally carry information using other table_ids. When acquiring and monitoring for DSM-CC object carousel sections, terminals shall silently ignore table_ids not supported for carriage of DSM-CC object carousel information.

NOTE: The present document only requires support for table_id 0x3b, 0x3c or 0x3d as defined in ISO/IEC 13818-6 [i.12].

The terminal shall consider cached information to remain valid only whilst the relevant object carousel is mounted and is being monitored. This prevents the possibility of retrieving stale data from a carousel which has been unmounted and remounted if the version number of an object has been incremented such that it has the same value as when it was cached. For the avoidance of doubt, changes to DSI messages shall not be considered to be an unmounting of the carousel.

The terminal shall consider cached information to remain valid only whilst the relevant PMT that signals the carousel is being monitored. The cache ceases to be valid if the carousel signalling is removed from the PMT.

The validity of any cached information is dependent only on the relevant object carousel and is independent of the lifecycle of any application, including applications delivered within that carousel.

Any cached information that is invalid shall be flushed from the cache.

The cache ceases to be valid when the selected broadcast service changes unless the new service contains the same carousel as the previous service (see clause B.2.10 of TS 102 809 [3]) and the terminal is able to monitor the carousel continuously.

Nothing in this clause shall affect the operation of the File System Acceleration persistent store (see clause 7.2.7).

When the CICAM Auxiliary File System is implemented as specified in the DVB Extensions to CI Plus TS 103 205 [37] and the HbbTV® Application Domain is offered by the CICAM, the terminal shall be able to request an application from this file system, as specified in annex F of TS 103 205 [37]. In this method the URL for the application is retrieved from the `simple_application_location_descriptor`.

7.2.3 Signalling of applications

7.2.3.1 Broadcast signalling

Table 5 identifies the descriptors and other signalling entities whose MPEG-2 encoding shall be supported. Clause numbers and page numbers refer to TS 102 809 [3].

Terminals shall support AIT subtables for HbbTV® applications, i.e. that have an application type 0x10, with at least 8 sections.

Elementary streams that are used to carry an application information table may additionally carry information using other table_ids. When acquiring and monitoring for AIT elementary streams, terminals shall silently ignore table_ids not supported for carriage of AIT information.

NOTE: The present document only requires support for table_id 0x74 as defined in TS 102 809 [3].

Table 5: Supported application signalling features

Clause	Page	Status	Notes
5.2.2 Application types	14	M	The application type shall be 0x0010.
5.2.3 Application identification	15	M	The value of the application_id has no significance for whether an application is trusted or not - see clause 11.1 for more information.
5.2.4 Application control codes	16	M	The following control codes shall be supported: 0x01 AUTOSTART 0x02 PRESENT 0x04 KILL 0x07 DISABLED The application life cycle shall follow the rules defined in TS 102 809 [3] and in the present document.
5.2.5 Platform profiles	17	M	For applications that only require the basic profile, the application_profile shall take the value 0x0000.The following bits can be combined to express profiles corresponding to additional features that applications may require: 0x0001 A/V content download feature 0x0002 PVR feature The 3 most significant bits of the application_profile are reserved for future use. As defined in clause 5.2.5.1 of TS 102 809 [3], terminals shall be able to run all applications where the signalled application profile is one of the profiles supported by the terminal. All terminals shall support the basic profile (0x0000) in addition to profiles corresponding to the other features supported by the terminal. The version fields shall be set as follows: version.major = 1 version.minor = 4 version.micro = 1 Additionally terminals shall launch applications signalled with the following values for major, minor and micro – [1,1,1], [1,2,1] and [1,3,1] – and run them as defined by the requirements in the present document.
5.2.6 Application visibility	18	See the Notes column	VISIBLE_ALL shall be signalled. Values other than VISIBLE_ALL are not included in the present document.
5.2.7 Application priority	18	M	
5.2.8 Application icons	19	O	The icon locator information shall be relative to the base part (constructed from the URL_base_bytes) of the URL as signalled in the transport_protocol_descriptor.
5.2.9 Graphics constraints	21	NI	
5.2.10 Application usage	22	M	Usage type 0x01 shall be supported as described in clauses 5.3.4 and 6.
5.2.11 Stored applications	23	NI	
5.2.12 Application Description File	26	NI	
5.3.2 Program specific information	28	M	

Clause	Page	Status	Notes
5.3.4 Application Information Table	29	M	A maximum of one PID per service shall be used to carry the AIT subtable defined by the HbbTV® application type. All sections of the HbbTV® AIT subtable shall be transmitted at least once every second. Terminals shall ignore AIT subtables within the selected service which have an `application_type` that the terminal cannot decode.
5.3.5.1 Application signalling descriptor	33	M	If more than one stream is signalled in the PMT for a service with an `application_signalling_descriptor`, then the `application_signalling_descriptor` for the stream containing the AIT for the HbbTV® application shall include the HbbTV® `application type` (0x0010).
5.3.5.2 Data broadcast id descriptor	33	O	The value to be used for the `data_broadcast_id` field of the `data_broadcast_id_descriptor` for HbbTV® carousels shall be 0x0123. The `id_specific_data` are not defined. By supporting this optional feature, terminals can reduce the time needed to mount a carousel.
5.3.5.3 Application descriptor	34	M	
5.3.5.4 Application recording descriptor	35	M/NI	Support of the `application_recording_descriptor` is mandatory when the terminal has support for time-shift. Otherwise it is not included. The semantics of the `application_recording_descriptor` for HbbTV® is clarified below this table.
5.3.5.5 Application usage descriptor	37	M	Usage type 0x01 shall be supported as described in clauses 5.3.4 and 6.
5.3.5.6 User information descriptors	38	M	
5.3.5.7 External application authorization descriptor	39	NI	
5.3.5.8 Graphics constraints descriptor	39	NI	
5.3.6 Transport protocol descriptors	40	M	The following `protocol_ids` shall be supported: 0x0001 object carousel over broadcast channel 0x0003 HTTP over back channel (i.e. broadband connection). The `protocol_id` 0x0004 CICAM Auxiliary File System shall be supported when the CICAM Auxiliary File System is implemented as specified in the DVB Extensions to CI Plus TS 103 205 [37]. When the `protocol_id` is 0x0003, only the simplified form (as defined in TS 102 809) shall be supported.
5.3.7 Simple application location descriptor	43	M	When the `protocol_id` is 0x0004, the application location descriptor shall reference an initial object provided by the CICAM. The domain identifier shall not be included, but shall be derived from the application type field.
5.3.8 Simple application boundary descriptor	43	M	Only strict prefixes starting with "`dvb://`", "`http://`", "`https://`", or "`ci://`" shall be supported. Only prefixes forming at least a second-level domain shall be supported. Path elements shall be ignored.
5.3.9 Service information	44	M	As modified by clause 7.2.6.
5.3.10 Stored applications	46	NI	

Table 6: Key to status column

Status	Description
M	MANDATORY The terminal shall support the referenced signalling. The signalling may be restricted to a subset specified in the "Notes" column. In that case all additional signalling is optional.
O	OPTIONAL It is the manufacturer's decision to support the referenced signalling.
NI	NOT INCLUDED The referenced signalling is not included in the present document. It should not be implemented unless required by another specification.

The semantics of the `application_recording_descriptor` are as follows:

- Applications that are safe to run in time-shift including trick mode shall set the `trick_mode_aware` flag and the `time_shift_flag` to '1'.
- The `scheduled_recording_flag` is not included.
- If applications are signalled with `trick_mode_aware` set to '0' the `time_shift_flag` shall be ignored.
- The `dynamic_flag` and `av_synced_flag` shall be used as defined by TS 102 809 [3].
- `initiating_replay_flag` is not included.
- `label_count`, `label_length`, `label_char` and `storage_properties` are not included.
- Applications shall list broadcasted data components in the component tag list. The elementary stream carrying the AIT does not need to be listed.

In addition, the broadcast AIT may contain a `parental_rating_descriptor`, as defined in EN 300 468 [16], carried in the "application" (inner) descriptor loop of the AIT. Terminals shall support this descriptor, as defined in clause 6.2.2.10.

When the CICAM Auxiliary File System is implemented, a CICAM application can be signalled as specified in annex F of the DVB Extensions to CI Plus TS 103 205 [37].

7.2.3.2 Broadcast-independent application signalling

The present document does not define any signalling, announcement or discovery of broadcast-independent applications. Clause 5.3.5 of the present document defines how they can be started. Broadcast-independent applications shall be identified either by the URL of the first page of the application or by the URL of a XML AIT as defined in clause 5.4 of TS 102 809 [3] and profiled in this clause. The XML file shall contain an application discovery record containing one or more `<application>` elements, all with the same orgId and appId values but with different application types. The XML file shall be delivered with HTTP or HTTPS using the "`application/vnd.dvb.ait+xml`" MIME type as defined in clause 5.4 of TS 102 809 [3].

The semantics of the fields and elements in the XML AIT file shall be as defined in Table 7.

Table 7: Contents of XML AIT for Broadcast-independent applications

Field or element	Requirement on XML AIT file	Requirement on terminal
appName	Optional.	Optional for terminal to use.
applicationIdentifier	Mandatory.	Mandatory.
applicationDescriptor/ type/OtherApp	Shall be "application/vnd.hbbtv.xhtml+xml" for HbbTV® applications. See note.	Mandatory. MIME types other than "application/vnd.hbbtv.xhtml+xml" are outside the scope of the present document.
applicationDescriptor/ controlCode	Shall be AUTOSTART.	Values other than AUTOSTART are outside the scope of the present document.
applicationDescriptor/ visibility	Shall be VISIBLE_ALL.	Values other than VISIBLE_ALL are outside the scope of the present document.
applicationDescriptor/ serviceBound	Shall be false.	Values other than false are outside the scope of the present document.
applicationDescriptor/ priority	Shall be present.	No defined semantics in the present document.
applicationDescriptor/ version	Outside the scope of the present document.	Outside the scope of the present document.
applicationDescriptor/ mhpVersion	Shall be the same values as defined for the MPEG-2 encoding of the AIT under "platform profiles" in Table 5.	Values higher than those defined in Table 5 shall result in the application failing to start.
applicationDescriptor/ icon	Optional.	Optional for terminal to use.
applicationDescriptor/ storageCapabilities	Outside the scope of the present document.	Outside the scope of the present document.
applicationTransport/	Mandatory. Shall be HTTPTransportType.	Mandatory.
applicationLocation/	Mandatory.	Mandatory.
applicationBoundary/	Optional.	Mandatory. Only strict prefixes starting with "dvb://", "http://","https:// " or "ci://" shall be supported.Only prefixes forming at least a second-level domain shall be supported. Path elements shall be ignored.
applicationSpecificDescriptor	Optional	Outside the scope of the present document
applicationUsageDescriptor	Outside the scope of the present document.	Outside the scope of the present document.
NOTE: This value shall be used in the XML AIT regardless of whether the application uses HTML or XHTML serialization, or whether it was authored for a previous revision of the present document. See also clause A.2.6.		

Where a value, element or attribute is indicated as being outside the scope of the present document, the presence of this value, element or attribute in an XML AIT is not prohibited but the present document does not require any behaviour from terminals other than not suffering from a fatal error and continuing to parse the remainder of the XML AIT.

When parental ratings are required, the XML AIT format described in clause 5.4 of TS 102 809 [3] is extended to provide parental rating information for broadcast-independent applications. The inclusion of a <ParentalRating> element as defined below in the extended format of the application's <ApplicationDescriptor> element indicates the parental rating for that application. The interpretation of <ParentalRating> is defined in the OIPF DAE specification [1] section E.3, as clarified in clause A.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:hbbtv="urn:hbbtv:application_descriptor:2014" xmlns:ait="urn:dvb:mhp:2009"
           xmlns:oipf="urn:oipf:iptv:ContentAccessDownloadDescriptor:2008-1"
           targetNamespace="urn:hbbtv:application_descriptor:2014"
           elementFormDefault="qualified" attributeFormDefault="unqualified" >
    <xs:import namespace="urn:dvb:mhp:2009" schemaLocation="oipf/imports/mis_xmlait.xsd"/>
<xs:import namespace="urn:oipf:iptv:ContentAccessDownloadDescriptor:2008-1"
               schemaLocation= "oipf/iptv-ContentAccessDownloadDescriptor.xsd"/>
    <xs:complexType name="HbbTVApplicationDescriptor">
        <xs:complexContent>
            <xs:extension base="ait:ApplicationDescriptor">
                <xs:sequence>
                    <xs:element name="ParentalRating" type="oipf:ParentalRatingType"
```

```
                                minOccurs="0" maxOccurs="unbounded" />
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

An example of an XML AIT using this schema (informative):

```
<?xml version="1.0" encoding="UTF-8"?>
<mhp:ServiceDiscovery
    xmlns:mhp="urn:dvb:mhp:2009"
    xmlns:hbb="urn:hbbtv:application_descriptor:2014">

    <mhp:ApplicationDiscovery DomainName="example.com">
        <mhp:ApplicationList>
            <mhp:Application>
                <mhp:appName Language="eng">Whizzo Play Along Quiz</mhp:appName>
                    <mhp:applicationIdentifier>
                    <mhp:orgId>123</mhp:orgId>
                    <mhp:appId>456</mhp:appId>
                </mhp:applicationIdentifier>
                <mhp:applicationDescriptor xsi:type="hbb:HbbTVApplicationDescriptor">
                    <mhp:type>
                        <mhp:OtherApp>application/vnd.hbbtv.xhtml+xml</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>01</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:ParentalRating Scheme="dvb-si" Region="GB">8</hbb:ParentalRating>
                </mhp:applicationDescriptor>
                <mhp:applicationTransport xsi:type="mhp:HTTPTransportType">
                    <mhp:URLBase>https://www.example.com/</mhp:URLBase>
                </mhp:applicationTransport>
                <mhp:applicationLocation>whizzo-app.html?a=1</mhp:applicationLocation>
            </mhp:Application>
        </mhp:ApplicationList>
    </mhp:ApplicationDiscovery>
</mhp:ServiceDiscovery>
```

7.2.4 Synchronization

The terminal shall support "do-it-now" events as defined in clause 8 of TS 102 809 [3]. Support of events synchronized to a DVB timeline as referred to in that document is not included.

Broadcasters shall place all "do-it-now" stream descriptors to be monitored simultaneously by an application on a single PID. This may be the same PID as is used for other DSM-CC sections.

7.2.5 DSM-CC carousel

7.2.5.1 Mounting related constraints

A terminal shall mount a maximum of one carousel at a time for use by the running application. Mounting means that the terminal makes the latest version of the files of the carousel available to the application. Additionally a terminal may read, cache and monitor several carousels in parallel in order to decrease the loading time as experienced by the user.

Terminals shall support carousels split across up to and including three elementary streams simultaneously as defined in clause 10.2.1.

NOTE: Typically, mounting a carousel may involve reading data from the carousel into a cache and monitoring for updates to the carousel.

7.2.5.2 Initial carousel mounting

A broadcast-related application whose initial page is broadcast will cause its carousel to be mounted by the terminal (in order to be loaded and launched) unless mounting the carousel would require tuning to a transport stream other than the one carrying the current channel. If tuning would be required, the attempt to load the page shall fail as if the file did not exist.

A broadcast-related application whose initial page is not broadcast may mount a carousel on the same service using the `component_tag`, e.g. through an `XMLHttpRequest` request or a reference (e.g. from an `<img>` element). If the elementary stream pointed to by the `component_tag` does not contain a service gateway, the mounting will fail.

The terminal shall not allow broadcast-independent applications to mount carousels. In order to mount a carousel or access any other broadcast resources, a broadcast-independent application will have to first become a broadcast-related application (see clause 6.2.2.6).

7.2.5.3 Subsequent carousel mountings (during the lifecycle of an application)

For a broadcast-related application, once a carousel has been mounted, a request that would require another carousel to be mounted shall succeed and cause the previous carousel to be un-mounted and all of its pending requests to be cancelled, unless mounting the carousel would require tuning to a transport stream other than the one carrying the current channel.

7.2.5.4 Constraints

A resolved DSM-CC object reference shall be at most 64 bytes.

7.2.6 Data services

HbbTV® services may exist that do not have any broadcast audio or video components (i.e. pure data services). Their broadcast signalling shall be as follows.

The SDT entry for the pure data service shall use a `service_descriptor` with a service_type of 0x0C. It shall also contain a `data_broadcast_descriptor` as defined in TS 102 809 [3] clause 5.3.9.1 with the following restrictions:

- The `data_broadcast_id` shall be 0x0123.
- The `selector_bytes` shall be present, and shall carry information about all HbbTV® AUTOSTART applications that the service may carry.
- The application name and text and other private data may be present.

The signalling of the AIT and any HbbTV® carousel remains the same as normal audio and video services.

Terminals shall process the `data_broadcast_descriptor` in the SDT and include, in the terminals service list, data services that signal applications that are supported. If the `selector_bytes` are not present, the service shall not be included in the terminals service list.

NOTE: The present document does not contain any requirements how broadcast channel lists are updated and managed. These requirements may be defined by the appropriate specifications for each market where the terminals are to be deployed.

Where an instance of the `Channel` class represents a data service, the value of the `channelType` property shall be 256.

7.2.7 File system acceleration

7.2.7.1 Introduction

Terminals shall support File System Acceleration (FSA) as defined by ES 202 184 [36] and further profiled in this clause.

7.2.7.2 HbbTV® stored groups descriptor

HbbTV® profile of File System Acceleration replaces the `stored_groups_descriptor` defined by ES 202 184 [36], clause 11.17.3 with the HbbTV® stored group descriptor as defined in Table 8.

Table 8:Syntax of the HbbTV® stored group descriptor

Syntax	Bits	Type
hbbtv_stored_groups_descriptor {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for (j=0;j<N;j++) {		
organisation_id	32	uimsbf
group_id	16	uimsbf
group_priority	8	uimsbf
use_from_carousel	1	bslbf
reserved	7	bslbf
application_profile	16	uimsbf
version_major	8	uimsbf
version_minor	8	uimsbf
version_micro	8	uimsbf
group_version	8	uimsbf
private_data_length	8	uimsbf
for (k=0;k<M;k++) {		
private_data_bytes	8	uimsbf
}		
}		
}		

The semantics are as defined in ES 202 184 [36] with the following exceptions:

- descriptor_tag: 0x81.
- organisation_id: equivalent to the `organisation_id` as defined in TS 102 809 [3].
- application_profile: equivalent to the `application_profile` as defined in TS 102 809 [3].
- version_major: equivalent to the `version.major` as defined in TS 102 809 [3].
- version_minor: equivalent to the `version.minor` as defined in TS 102 809 [3].
- version_micro: equivalent to the `version.micro` as defined in TS 102 809 [3].

A terminal shall only cache a group if it supports the profile identified by the `application_profile`, `version_major`, `version_minor` and `version_micro` fields.

7.2.7.3 Group location descriptor

The `group_location_descriptor` defined in Table 11.73 of ES 202 184 [36] may be included in the private data bytes of the `hbbtv_stored_groups_descriptor` to define a Group Location other than the default of "`DSM:/`". Terminals are not required to support Group Locations not in the current object carousel. File Groups with Group Locations that do not begin with "`DSM:/`" should not be supported unless required by another specification.

7.2.7.4 Group Manifest file name

HbbTV® profile of File System Acceleration redefines the Group Manifest file name as follows:

`/<organisation_id>-<group_id>.man`

Where `<organisation_id>` and `<group_id>` are the values from the `hbbtv_stored_groups_descriptor`. The `organisation_id` is a 32 bit value and represented as 8 hexadecimal chars with lower case characters. The `group_id` is a 16 bit value and represented as 4 hexadecimal chars with lower case characters. For example, a group with `organisation_id` = 1 and `group_id` = 12 shall have a group manifest file name of:

```
/00000001-000c.man
```

7.2.8 Protocol for download

Where content download is supported, the FDP protocol, defined in annex H, shall be supported.

A/V content downloaded via FDP shall satisfy the same requirements as A/V content downloaded via broadband, as specified in clause 7.3.1.

7.3 Broadband-specific format and protocols

7.3.1 System, video and audio formats

7.3.1.1 General requirements

The system formats and their labels are specified in the OIPF Media Formats specification [2] with the restrictions in clause 7.3.1.2.

The video formats and their labels are specified in the OIPF Media Formats specification [2] with the restrictions and extensions in clause 7.3.1.3.

The audio formats are specified in the OIPF Media Formats specification [2] with the restrictions in clause 7.3.1.4 with the addition of non-interleaved IEEE 32-bit linear PCM as defined in clause 2.9 of Web Audio [65].

The subtitle formats are specified in clause 7.3.1.5.

The subtitle format EBU-TT-D is specified in the EBU-TT-D specification [43] with the restrictions in clause 7.3.1.5. For content based on DVB transport stream the terminal shall support the same subtitle formats (referred to as BCSUB) for content received by the broadband connection as are supported for the broadcast connection. The terminal shall support out-of-band EBU-TT-D subtitles regardless of the underlying system format.

Table 9 defines the subset of the combinations of system, video, audio and subtitle formats specified in the OIPF Media Formats specification [2] and the present document that shall be supported for non-adaptive HTTP streaming and with the download option.

Table 9: System, video and audio formats for non-adaptive HTTP Streaming and content download

System Format	Video Format	Audio Format	Subtitle format (see note 4)	MIME Type
TS	AVC_SD_25 AVC_HD_25	HEAAC E-AC3 (see note 1)	(see note 2)	video/mpeg
MP4	AVC_SD_25 AVC_HD_25 HEVC_HD_25_8 (see note 3) HEVC_HD_25_10 (see note 3) HEVC_UHD_25 (see note 3)	HEAAC E-AC3 (see note 1)	EBU-TT-D (see note 7)	video/mp4
-	-	MPEG1 L3	-	audio/mpeg
MP4	-	HEAAC (see note 5) E-AC3 (see note 1)	-	audio/mp4
TS	-	HEAAC E-AC3 (see note 1)	-	audio/mpeg
TS	-	-	(see notes 2 and 6)	image/vnd.dvb.subtitle

NOTE 1: Terminals shall support E-AC3 for content received by the broadband connection when it is supported for the broadcast connection. Otherwise it is not mandated.
NOTE 2: Terminals shall support the same subtitle formats for content received by the broadband connection as are supported for the broadcast connection. See clause 7.3.1.5.2.
NOTE 3: Only applicable to terminals that support HEVC as described in clause 7.3.1.3.
NOTE 4: Terminals shall support out-of-band subtitles regardless of the underlying system format.
NOTE 5: This is carriage of HE-AAC audio inside the MP4 system format container. This format shall comply with the requirements specified in clause 8.6.35 of the DLNA media formats specification IEC 62481-2 [26], except for clause 8.6.35.11.
NOTE 6: This format definition is intended for the use of multi-stream synchronization as defined in clause 10.2.8. Terminals are not required to support this format for any other use case.
NOTE 7: Support for video, audio and inband subtitles multiplexed into a single ISOBMFF file is only required for downloaded content. It is not required for non-adaptive HTTP streaming.

For MPEG DASH, the following shall apply;

- AVC_SD_25 and AVC_HD_25 shall be supported. The conditions on support for HEVC_HD_25_8, HEVC_HD_25_10, and HEVC_UHD_25 in clause 7.3.1.3 shall apply.
- HE-AAC shall be supported. The conditions on support for E-AC3 in Table 9 shall apply.

The only system format required for MPEG DASH in the present document is ISOBMFF. The only subtitle format required for MPEG DASH in the present document is EBU-TT-D. MIME types are addressed in the DVB DASH profile TS 103 285 [45].

Playing WAVE audio from memory is not included in the present document. It should not be implemented unless required by another specification.

Examples of media which comply with the above supported codecs list:

- "`http://myserver/myvideo.mp4`", mimetype "`video/mp4`", container "`mp4`", 2.5 MBit/s, resolution 720 × 576 @ 25 frames per second, together with AAC LC sound @ 64 kBit/s.
- "`http://myserver/myaudio.mp3`", mimetype "`audio/mpeg`", container "`mp3`", 256 kBit/s.

7.3.1.2 Systems layers

The usage of the systems layer format MPEG-2 Transport Stream shall comply with clause 4 of the OIPF Media Formats specification [2]. Support for the DLNA extension "time stamped MPEG-2 transport stream" is not required. Support for EIT, object carousel and 3D is not required. Support for AIT signalling is not required and it shall not be processed if it is present. See clause 13.4.2 for requirements to support the TEMI timeline for this system layer format.

The MP4 File Format shall comply with clause 4 of the OIPF Media Formats specification [2] and the following additions:

- The size of the moov box should not exceed 2.5 MByte. The requirement in OIPF that "The size of the moov box shall be equal to or less than 2Mbytes" does not apply in the present document.

  NOTE 1: Large moov boxes will slow down start up times especially for broadband connections with asmall bandwidth.

- The size of a box should not exceed 4 GByte.

Support for media zone information is not required in any system layer.

For non-adaptive unicast streaming, terminals shall support content whose average total bitrate when measured over a 10 second window does not exceed 12 Mbit/s, inclusive of multiplexing and packaging overheads but excluding network protocol overheads.

For adaptive bitrate streaming, terminals shall support combinations of one video, one audio and one subtitle representation for which both of the following conditions are satisfied:

- the combined channel bandwidth requirement for continuous playback does not exceed:
  - 12 Mbit/s if the terminal does not support UHD video.
  - 26 Mbit/s if the terminal does support UHD video.
- the combined channel bandwidth of the Representations whose media segments are delivered using TLS does not exceed 12 Mbps. This requirement shall apply for all TLS cipher suites that the terminal offers when establishing a connection for the purposes of media delivery. For requirements on TLS cipher suites, see clause 11.2.2.

NOTE 2: For MPEG DASH content, the channel bandwidth requirement for continuous presentation of a Representation is indicated in the Representation@bandwidth attribute in the MPD. It covers packaging overheads but does not include protocol overheads.

Adaptive bitrate presentations may include representations which exceed these rates, subject to the maximum bitrate limitations of the codecs being used. Terminals may ignore representations with bitrate requirements greater than the above limits and shall ignore representations with bitrate requirements that exceed the terminal's capabilities.

7.3.1.3 Video

The video format AVC_SD_25 shall comply with clauses 5.1.2.1 and 5.1.6 of the OIPF Media Formats specification [2].

The video format AVC_HD_25 shall comply with clauses 5.1.1.1 and 5.1.6 of the OIPF Media Formats specification [2].

NOTE 1: Terminals do not have to support non-adaptive HTTP streaming content where the following video parameters change: frame rate, interlaced / progressive, resolution, codec profile or level, colour space.

In addition:

- Terminals that support 8-bit HEVC HD video on the broadcast connection defined by DVB as "50 Hz HEVC HDTV 8-bit IRD" in TS 101 154 [14] shall also support 8-bit HEVC HD for content received by the broadband connection.

  Such content corresponds to the video format label HEVC_HD_25_8.

NOTE 2: Since terminals that support 10-bit HEVC also support 8-bit HEVC, applications should search for HEVC_HD_25, not HEVC_HD_25_8.

Terminals that support 10-bit HEVC HD video on the broadcast connection defined by DVB as "50 Hz HEVC HDTV 10-bit IRD" in TS 101 154 [14] shall also support 10-bit HEVC HD for content received by the broadband connection.

Such content corresponds to the video format label HEVC_HD_25_10.

Terminals shall include at most one of the format labels HEVC_HD_25_8 or HEVC_HD_25_10 in the XML capabilities.

- Terminals that support HEVC UHD video on the broadcast connection defined by DVB as "HEVC UHDTV IRD" in clause 5.14.3 of TS 101 154 [14] shall also support HEVC UHD for content received by the broadband connection. Such content corresponds to the video format label HEVC_UHD_25.

NOTE 3: There is no requirement to support the frame rates required by the 60 Hz HEVC HDTV IRDs as required by clause 5.14.1.7 of TS 101 154 [14].

Different requirements on video resolutions and decoder capabilities apply to content delivered using MPEG DASH as defined in annex E.

7.3.1.4 Audio

Audio formats shall comply with clause 8.1 of the OIPF Media Formats specification [2] with the following additional requirements for multichannel audio:

- If the terminal supports a stereo output, it shall be capable of providing a down-mix of multichannel audio to stereo.

- If the terminal is equipped with a digital audio output then it shall be capable of providing the bitstream at this output (pass-through) and should be capable of transcoding multi-channel audio from HEAAC to AC3 format.

- The terminal shall use metadata, where provided, to control the stereo down-mix from multichannel audio, and shall use it, or pass it through, when providing bitstream output. Such metadata may be provided as described in the OIPF Media Formats specification [2] and clause 6.8 of TS 102 366 [15].

- The remaining text in this clause applies to normal transcoding and does not consider cases where application or system sounds are inserted in addition.

  If AC-3 is used to output audio over S/PDIF, HDMI or similar transfer protocols, terminals shall transcode the available metadata of an incoming HE-AAC or E-AC-3 audio stream to match the constraints of the AC-3 bit stream syntax.

  Incoming metadata parameters with values exceeding the range or granularity of the corresponding parameters in AC-3 shall be rounded to the closest value creating a lower audio output level where possible to meet the range and granularity limitations of the AC-3 bit stream syntax.

  The metadata transformed in order to meet the limitations of the subsequent AC-3 audio format may also be applied on the local PCM outputs of a receiver. Potential side-effects of such proceeding e.g. an impact on artistic intent should be carefully considered.

  Examples for mapping of parameters:

  1. HE-AAC prog_ref_level of -21.75dB mapped to AC-3 dialnorm of -21dB

  2. HE-AAC prog_ref_level of -31.75dB mapped to AC-3 dialnorm of -31dB

  3. E-AC-3 lorocmixlev of -∞dB mapped to AC-3 cmixlev of -6dB

  4. E-AC-3 lorosurmixlev of -4.5dB mapped to AC-3 surmixlev of -6dB

  If the AC-3 encoder supports Annex D of TS 102 366, E-AC-3 downmix coefficients are fully supported. HE-AAC downmix coefficients may be mapped to lorocmixlev and lorosurmixlev.

  The AC-3 metadata parameters ltrtcmixlev and ltrtsurmixlev as defined in Annex D of TS 102 366 have no corresponding parameters in HE-AAC. If the AC-3 encoder supports Annex D of TS 102 366 the default value for ltrtsurmixlev and ltrtcmixlev is -3 dB.

  Legacy AC-3 decoders that do not support Annex D of TS 102 366 ignore lorocmixlev/lorosurmixlev and ltrtcmixlev/ltrtsurmixlev and use cmixlev/surmixlev instead.

7.3.1.5 Subtitles

7.3.1.5.1 TTML based subtitles

Terminals shall be able to correctly render TTML based subtitles with the following constraints:

- The subtitle document shall be compliant with the EBU-TT-D specification [43] (referred to as "EBU-TT-D" format).
- The subtitle document shall have no more than 8 concurrently visible regions.
- The subtitle document shall use UTF-8 character encoding.

Terminals shall support the `ebutts:multiRowAlign` and `ebutts:linePadding` extensions to TTML defined in the EBU-TT-D specification [43].

On terminals that support EBU-TT-D version 1.0 and no later version, where the `tts:lineHeight` attribute of a `tt:p` element has the value "`normal`" or a value less than or equal to 125%, the background of each generated inline area shall be rendered such that there are no gaps between the rendered backgrounds of adjacent lines.

Reception of EBU-TT-D subtitles shall be supported in-band in the following ways:

- with MPEG DASH content as defined in annex E and the DVB DASH profile TS 103 285 [45];
- (if the download option is supported) with ISOBMFF content as defined by the EBU specification for carrying EBU-TT-D in ISOBMFF [44] that has been downloaded by the terminal regardless of whether the content was downloaded via broadband or via broadcast (using FDP as defined in annex H of the present document).

NOTE 1: The present document does not require support for non-adaptive HTTP streaming of a single ISOBMFF file with video, audio and subtitles all interleaved.

Support for in-band EBU-TT-D subtitles shall be indicated in the XML capabilities (see clause 10.2.4) using "`EBUTTD`" as the subtitle format name.

Out of band delivered EBU-TT-D subtitles shall be supported irrespective of how the A/V content is being or has been delivered. In this case terminals shall support EBU-TT-D subtitles contained in a single XML document delivered via HTTP, with a document size of up to and including 512kByte.

NOTE 2: If applications need larger subtitle documents they should use MPEG DASH format.

NOTE 3: When used with audio-only content, the subtitle plane still follows the size of the video plane even though there is no video.

Terminals shall support EBU-TT-D subtitle content that has associated downloadable fonts. Terminals shall be able to download and use at least one downloadable font, in addition to any resident fonts, when rendering a subtitle stream. The font formats required by the DVB DASH specification TS 103 285 [45] shall be supported.

When resolving `tts:fontFamily` references from EBU-TT-D subtitles, terminals shall search for a match in fonts that have been successfully downloaded before considering the embedded fonts listed in clause 10.2.1. When matching embedded fonts, the following mappings for TTML `genericFamilyName` shall apply:

- "`default`", "`sansSerif` " and "`proportionalSansSerif`" shall match the Tiresias™ embedded font;
- "`monospace`" and "`monospaceSansSerif`" shall match the Letter Gothic embedded font.

Other `genericFamilyName`s may match an appropriate embedded font if one is available; otherwise they shall be treated as "`default`".

In the case of MPEG DASH content, terminals shall observe the signalling of downloadable fonts defined in the DVB DASH specification TS 103 285 [45]. Fonts shall be downloaded when referenced using either an `EssentialProperty` or a `SupplementalProperty`. Error handling shall be as defined in the DVB DASH specification TS 103 285 [45].

Terminals shall support downloadable fonts signalled in the Content Access Streaming Descriptor for streaming content as defined in clause 7.3.2.1 and if the download option is supported signalled in the Content Access Download

Descriptor for download content as defined in clause 7.3.2.2. Terminals shall support the extensions of the descriptors for downloadable fonts as defined in clause A.2.25.

If a terminal is unable to download a font for any reason or having downloaded a font is unable to use it, then:

- If the font download has the `@essential` attribute set to `true`, the terminal shall not present subtitles for the stream referenced by the descriptor.
- If the font download does not have the `@essential` attribute set to `true`, the terminal shall present the subtitles as if this `<DownloadableFont>` element were not included.

7.3.1.5.2 Broadcast subtitles

As defined in clause 7.3.1.1 above, terminals shall support the same subtitle formats for MPEG-2 transport stream content received by the broadband connection as are supported for the broadcast connection. Specifically:

- DVB subtitles
- EBU teletext subtitles

Basic specification references and labels for both are specified in the OIPF Media Formats specification [2] although the base specifications may be subject to modification, clarification and profiling in broadcast channel specifications for particular markets.

If supported in general as defined above terminals shall support broadcast subtitles received via broadband as part of a SPTS, including:

- TV services.
- Radio services.
- Subtitle-only streams with DVB subitles synchronized with a broadcast service as defined in clause 10.2.8. The API for broadband delivered streams only carrying broadcast subtitle only streams is defined in clause A.2.5.4.
- Optionally, subtitle-only streams with EBU teletext subtitles synchronized with a broadcast service as defined in clause 10.2.8.

NOTE: Other encapsulations or pure elementary streams are not required to be supported for broadcast subtitles.

7.3.2 Protocols

7.3.2.1 Protocols for streaming

Unicast streaming using HTTP 1.1 shall be supported as defined in clause 5.3.2.2 of the OIPF protocols specification [4] with the addition that the `Content-Range` header shall be supported in seek operations thus allowing the application to seek to any arbitrary position within the streaming video without the need of downloading the complete video first. The terminal should only buffer data equivalent to approximately 10 seconds of normal play in advance of the current play position unless the download rate is consistently lower than the consumption rate. If the `Content-Length` header is not provided terminals shall not make any assumptions on the size of the buffer on the server. Hence terminals which need to obtain some data from the stream, e.g. for initialization, cannot assume that this data is still buffered on the server once they have completed their initialization.

The accuracy of seeking to a particular point in time within an MPEG-2 transport stream is implementation dependent. Applications should avoid this except for small seeks relative to the current position in a stream that is already being played which are likely to be the least inaccurate. Seeking is likely to be more accurate in a constant bit-rate stream than a variable bit-rate one.

HTTP chunked transfer coding shall be supported as defined by clause 3.6.1 of IETF RFC 2616 [6].

NOTE 1: The preferred method for delivery of live content is using MPEG DASH. Use of HTTP chunked transfer encoding for video or audio may be removed in a future revision of the present document.

NOTE 2: Live content delivered using HTTP chunked transfer encoding is presented using the A/V Control object. There are no requirements for the video/broadcast object to present content delivered using HTTP.

HTTP adaptive streaming shall be supported using MPEG DASH as defined in annex E.

7.3.2.2 Protocols for download

Where content download is supported, HTTP shall be supported as defined in clause 5.3.4 of the OIPF protocols specification [4].

7.3.2.3 Void

Void

7.3.2.4 HTTP User-Agent header

All outgoing HTTP requests made on behalf of an HbbTV® application, and during the process of launching an HbbTV® application, shall include a `User-Agent` header using the syntax described in this clause.

NOTE: This does not apply to HTTP requests made by the MPEG DASH player or the DRM agent.

The `User-Agent` header shall include:

```
HbbTV/1.4.1 (<capabilities>; <vendorName>; <modelName>; <softwareVersion>; [<hardwareVersion>];
<familyName>; <reserved>)
```

Where:

- The `<capabilities>` field consists of zero or more concatenated HbbTV® option strings as defined in clause 10.2.4.
- The `<vendorName>`, `<modelName>`, `<softwareVersion>`, `<hardwareVersion>` fields are the same as defined in clause 7.3.3.2 of the OIPF DAE specification [1]. The `<hardwareVersion>` field is optional.
- The `<vendorName>` field shall reflect the consumer-facing make / brand of the terminal.
- The `<vendorName>`, `<softwareVersion>`, `<familyName>` combination, along with `<hardwareVersion>` where included, shall differ between terminals with significantly different implementations or performance.
- The `<modelName>` field should be representative of the consumer-facing model name to allow log messages to be matched up with user reported problems.
- The `<familyName>` field shall have the semantics defined in clause 7.3.3.2 of the OIPF DAE specification [1] but shall additionally be chosen so as to be globally unique. This shall be achieved either by prefixing with a reverse domain name of the organisation allocating the remaining portion of the `familyName`, or by using a version 4 UUID as the `familyName`, formatted as a simple string (i.e. without any `urn:uuid` prefix) [64].
- The `<reserved>` field is reserved for future extensions.

This `User-Agent` header may be extended with other implementation-specific information including other user agent information. In particular, it is recommended to include the browser user agent information.

IETF RFC 2616 [6] permits a `User-Agent` header to be split over multiple lines by including the sequence `CRLF 1*( SP | HT )`. The `User-Agent` header shall not include this sequence and shall only include linear whitespace matching the sequence `1*( SP | HT )`.

Valid examples of this syntax are:

```
User-Agent: HbbTV/1.4.1 (+PVR+DL; Sonic; 40VX700WDR; 1.32.455; 2.002; com.example.2016VX700; )
BKit/445.27.1
```

```
User-Agent: HbbTV/1.4.1 (;Sonic; VX600WDR; 1.14.0; ; dd5528b6-d0a9-40a8-acdb-21fa2eabeb2e;)
```

7.3.2.5 HTTP Redirects

HTTP redirects as defined in IETF RFC 2616 [6] in response to an HTTP request shall be supported as described in this clause.

- The terminal shall support responses with a status code of "301 Moved Permanently", "302 Found", "303 See Other" and "307 Temporary Redirect" by using the temporary URL given in the Location field.
- The terminal shall support at least 10 redirections for requests made from the browser, and 3 redirections for requests made by media player functions.
- Infinite loops shall be detected. Terminals shall not follow them indefinitely. This may be achieved by detecting more than 10 redirects from an original request.

For the avoidance of doubt, these requirements shall apply to both http: and https: URIs and redirects between these.

7.3.2.6 HTTP Caching

Terminals shall observe the caching rules defined in HTTP/1.1 [6] including support for issuing requests for cached content using the "If-Modified-Since" header (where a server provides a Last-Modified header) and the "If-None-Match" HTTP header (where a server provides an ETag header).

Terminals shall support an HTTP cache for content accessed by the browser. This cache is not required to persist across power cycles.

NOTE: This does not apply to HTTP requests made by the MPEG DASH player or the DRM agent.

7.3.2.7 Simultaneous HTTP connections

Terminals shall support at least two simultaneous HTTP connections for application content in addition to any connections required for HTTP-based media streaming. Specifically, an HTTP request to one server that takes time to complete shall not delay a second HTTP request to a different server if no other application-initiated HTTP requests (other than for media streaming purposes) are in progress.

For the avoidance of doubt, these requirements shall apply to HTTP requests using any combination of http: and https: URIs.

8 Browser application environment

8.1 DAE specification usage

The OIPF DAE specification [1] shall be supported as defined in annex A of the present document.

8.2 Defined JavaScript APIs

8.2.1 Acquisition of DSM-CC stream events

8.2.1.1 Adding and removing stream event listeners

The following additional methods on the video/broadcast object (as defined in the OIPF DAE specification [1]) shall be supported for synchronization to broadcast events as defined in clause 7.2.4.

```
void addStreamEventListener(String targetURL, String eventName,
                            EventListener listener)
```

Description	Add a listener for the specified DSM-CC stream event. When a broadcaster transmits an identical instance of the MPEG private data section carrying a stream event descriptor (including the version number), only one `StreamEvent` event shall be dispatched. When a broadcaster transmits different events using the same event name id (i.e. with different version numbers), one `StreamEvent` event shall be dispatched for each different stream event descriptor received. An event shall also be dispatched in case of error. Listeners can only be added while the video/broadcast object is in the Presenting or Stopped states. Calls to this function when the video/broadcast object is in other states shall have no effect. The terminal shall automatically unregister all listeners on the video/broadcast object in the following cases: • A transition to the Unrealized state (e.g. when becoming broadcast-independent). • A transition to the Connecting state that is due to a channel change. Listeners are not unregistered when transitioning to the Connecting state due to a transient error that does not result in a change of channel.	
Arguments	`targetURL`	The URL of the DSM-CC `StreamEvent` object or an HTTP or HTTPS URL referring to an XML event description file (as defined in clause 8.2 of TS 102 809 [3]) describing the event.
	`eventName`	The name of the event (of the DSM-CC `StreamEvent` object) that shall be subscribed to.
	`listener`	The listener for the event.

`void removeStreamEventListener(String targetURL, String eventName, EventListener listener)`		
Description	Remove a stream event listener for the specified stream event name.	
Arguments	`targetURL`	The URL of the DSM-CC `StreamEvent` object or an HTTP or HTTPS URL referring to an event description file describing the event.
	`eventName`	The name of the event (of the DSM-CC `StreamEvent` object) whose subscription shall be removed.
	`listener`	The listener for the event.

8.2.1.2 DSM-CC StreamEvent event

```
interface StreamEvent : Event {
    readonly attribute String      name;
    readonly attribute String      data;
    readonly attribute String      text;
    readonly attribute DOMString   status;

}
```

Properties	`name`	The name of the DSM-CC `StreamEvent`'s event.
	`data`	Data of the DSM-CC `StreamEvent`'s event encoded in hexadecimal. EXAMPLE: "`0A10B81033`" (for a payload 5 bytes long).
	`text`	Text data of the DSM-CC `StreamEvent`'s event as a string assuming UTF-8 as the encoding for the DSM-CC `StreamEvent`'s event. Characters that cannot be transcoded are skipped.
	`status`	Equal to "`trigger`" when the event is dispatched in response to a trigger in the stream or "`error`" when an error occurred (e.g. attempting to add a listener for an event that does not exist, or when a `StreamEvent` object with registered listeners is removed from the carousel). Circumstances under which an event shall be dispatched with an error status include: • the `StreamEvent` object pointed to by `targetURL` is not found in the carousel or via broadband; • the `StreamEvent` object pointed to by `targetURL` does not contain the event specified by the `eventName` parameter; • the carousel cannot be mounted; • the elementary stream which contains the `StreamEvent` event descriptor is no longer being monitored (e.g. due to another monitoring request or because it disappears from the PMT). Once an error is dispatched, the listener is automatically unregistered by the terminal.

8.2.2 Carousel objects access with XMLHttpRequest

In order to access the content of a carousel file, the `XMLHttpRequest` object can be used with the following constraints:

- Parameters passed to the `open()` method:
  - `method`: Shall be set to "GET".
  - `url`: Can be relative (to the location of the current page in the carousel's file system) or an absolute `dvb:` URL.
  - `async`: shall be set to `true`.
  - `user` and `password`: Ignored.
- `status`: Set to 200 when the DSM-CC object is found and to 404 if it cannot be accessed - e.g. the object is not present in the carousel or if the carousel has been unmounted (due to another request).
- `statusText`: implementation dependent.
- Headers are not relevant for carousel access:
  - Calls to `setRequestHeader()` are ignored.
  - `getResponseHeader()` shall return null and `getAllResponseHeaders()` shall return an empty string.
- Values of the `responseText` and `responseXML` properties are shown in Table 10.

Table 10: Values of the responseText and responseXML properties

DSM-CC object	URL example	responseText	responseXML
File	`/weather/data.xml`	Returns the "text response entity body" as defined in XMLHTTPRequest.	If the file has the extension "`.xml`", returns the "XML response entity body" as defined in XMLHTTPRequest. Otherwise, returns null.
Directory	`/weather`	Comma-separated list of names (File name, Stream Event name or Directory name) of all objects in the directory. These names shall not include path information.	null
Stream Event	`/weather/main/streamEvt1`	Comma-separated list of names of all events in the Stream Event object.	null

Examples of `dvb:` URLs that may be used with the `XMLHttpRequest` object are:

```
/weather/data.xml(absolute path from the root of the carousel of the current page)
../weather/data.xml (relative path to the current page)
dvb://1..1.B8/weather/data.xml (0xB8 is the component tag)
```

8.2.3 APIs for media synchronization

8.2.3.1 Introduction (Informative)

This clause defines the API for multi-stream synchronization as defined in clause 10.2.8 and the API for the inter-device synchronization with a media presentation on a second terminal or companion screens.

As the architecture for multi-stream and inter-device synchronization is basically the same, the `MediaSynchroniser` embedded object defined in clause 8.2.3.2 is the main object for both synchronization features. The sequence diagrams in clause 13.8 provide further indication how the API interacts with the architecture entities.

Example usage for multi-stream synchronization:

```
// this example shows the API usage for synchronising a broadcast channel with a DASH stream
// delivered over broadband using MPEG TEMI as an additional content timeline in the
// broadcast

var vba1; // holds a video/broadcast object
// timeline is MPEG TEMI on elementary stream with component tag 8 and timeline id 1
timeline_spec_vba1 = 'urn:dvb:css:timeline:temi:8:1';

var dasha1; // holds a DASH media object
timeline_spec_dasha1 = 'urn:dvb:css:timeline:mpd:period:rel:1000:0b71a';

// create the MediaSynchroniser using the broadcast service. Its timeline is used as
// the timeline for the MediaSynchroniser API
ms = oipfObjectFactory.createMediaSynchroniser();

ms.initMediaSynchroniser(vba1, timeline_spec_vba1);

// ... some xmlHTTPRequest to get correlation timestamps (e.g. from MRS) for the DASH stream related
//   to the timeline for the MediaSynchroniser API

timestamp_vba1_dasha1 = {'tlvMaster' : 12345, 'tlvOther' : 12445};

ms.addMediaObject(dasha1, timeline_spec_dasha1, timestamp_vba1_dasha1);

// the terminal now synchronizes the DASH media object to the broadcast service

// ... repeat with correlation timestamps to adjust for drift etc.
timestamp_vba1_dasha1 = {'tlvMaster' : 15345, 'tlvOther' : 16448};
ms.updateCorrelationTimestamp(dasha1, timestamp_vba1_dasha1);

// synchronisation can be stopped by removing the DASH object from the MediaSynchroniser
ms.removeMediaObject(dasha1);
```

Example usage for inter-device synchronization:

```
// this example shows inter-device synchronisation between two HbbTV terminals
//  one terminal is acting as the master terminal, the second one is acting as the slave
//  the slave role could be taken by a companion device. For 10.2.8.1
```

JavaScript snippets on master terminal:

```
// open app2app communication and wait for slave terminals or companion devices to connect

// application on a slave terminal connects through app2app comm
// applications agree to start inter device synchronisation


var vba1; // holds a video/broadcast object
timeline_spec_vba1 = 'urn:dvb:css:timeline:temi:8:1';

// create the MediaSynchroniser using the broadcast service whose timeline is used as
// the timeline of the MediaSynchroniser API, this step is the same as for intermedia sync
ms = oipfObjectFactory.createMediaSynchroniser();

ms.initMediaSynchroniser(vba1, timeline_spec_vba1);

// enable slave terminals to synchronise with media objects attached to the MediaSynchroniser
ms.enableInterDeviceSync();

// tell the slave application that we are ready via app 2 app

// causes any sync with other devices to stopped gracefully and prevents other devices to start sync
ms.disableInterDeviceSync();
```

JavaScript snippets on slave terminal:

```
// discovering terminals available
app.discoverTerminals(callback);

function callback (discoveredTerminals) {

    if (discoveredTerminals.length > 0) {
        // pick a terminal from the array of discovered ones
        // usually ask the user, but here we choose the first in the array
        var discoveredTerminal = discoveredTerminals[0];
        // get app2app endpoint and connect to the application running on the master
        var app2appBaseUrl = discoveredTerminal.X_HbbTV_App2AppURL;

        // do app 2 app comm and agree to start with inter-device sync.
    }
}

// the master tells us that it is ready for sync via app2app

// discovering the CSS-CII endpoint on the master
var css_ci_service_url = app.getServiceEndpoint(master_terminal_id, 'X_HbbTV_InterDevSyncURL');

// the CSS-CII endpoint of the master is connected to the MediaSynchroniser that started
//  the inter-device synchronisation and hence also the media object that is used to create
//   the MediaSynchroniser
ms = oipfObjectFactory.createMediaSynchroniser();
ms.initSlaveMediaSynchroniser(css_ci_service_url);

var dashb1; // holds a DASH media object
timeline_spec_dashb1 = 'urn:dvb:css:timeline:mpd:period:rel:1000:0b71a';

// ... some xmlHTTPRequest to get correlation timestamps (e.g. to MRS) of the DASH media object
//   to the timeline used by the MediaSynchroniser API on the master terminal
ms.addMediaObject(dashb1, timeline_spec_dashb1, timestamp_vba1_dashb1);

// begin synchronisation of media added to this slave MediaSynchroniser with the master terminal
ms.enableInterDeviceSync(callback2);

function callback2() {
    // inter-device sync is now enabled successfully

}
```

```
// later (once inter-device sync is enabled successfully)

// ... adjust correlation timestamps to keep in sync
ms.updateCorrelationTimestamp(dashb1, timestamp_vba1_dashb1);

// ... cause any sync with the master to stop gracefully
ms.disableInterDeviceSync();
```

8.2.3.2 The MediaSynchroniser embedded object

8.2.3.2.0 General

The terminal shall support a non-visual embedded object of type "`application/hbbtvMediaSynchroniser`" with the following JavaScript API. The terminal shall support one operable `MediaSynchroniser` embedded object. If an application creates multiple instances only the last one created needs to be operable. A `MediaSynchroniser` can be used for multi-stream synchronization and for inter-device synchronization. Depending on the initialization method used, inter-device synchronization will be performed in either the role of a master terminal or a slave terminal.

8.2.3.2.1 Properties

`function onError (Number lastError, Object lastErrorSource)`	
Description	The function that gets called when an error occurs for this `MediaSynchroniser` object. The terminal shall pass two arguments in the call. The first argument shall be the error code as defined in clause 8.2.3.2.4. The second argument shall be the media object that was the cause of the error or a string equalling the URL passed to `initSlaveMediaSynchroniser()` if the cause of the error is the master terminal or interaction with the master terminal. If the error was not caused by a media object or the master terminal or interaction with the master terminal, then this shall be `null`.

`function onSyncNowAchievable (Object mediaObject)`	
Description	The function shall be called by the terminal if it was previously not possible for the terminal to time presentation of this media to synchronize it to the master media but now it has become possible and the terminal has started to do so. The argument shall be the media object for which synchronized presentation timing can now be achieved. This function is only called (with a given media object passed as argument) after an earlier transient error of the MediaSynchroniser with error code 1 or 11 as appropriate (relating to the same media object).

`readonly Number lastError`	
Description	Shall be the code of the last error that occurred for this `MediaSynchroniser` object as defined in clause 8.2.3.2.4.

`readonly Object lastErrorSource`	
Description	Shall be the media object that was the cause of the last error; or a string equalling the URL passed to `initSlaveMediaSynchroniser()` if the cause of the last error is the master terminal or interaction with the master terminal. If the error was not caused by a media object or the master terminal or interaction with the master terminal, then this shall be `null`.

`readonly Number nrOfSlaves`	
Description	If this `MediaSynchroniser` is being used as the master for inter-device synchronization, it shall be the number of current web socket connections on the CSS-CII service endpoint provided by the terminal. Otherwise it shall be null.

`readonly Boolean interDeviceSyncEnabled`	
Description	Shall be true if and only if the terminal is currently a master terminal or a slave terminal.

readonly Number interDeviceSyncDispersion	
Description	The dispersion of clock synchronization between the slave terminal and the master terminal in milliseconds. This value quantifies the limit on the accuracy of synchronization currently achievable. If the terminal has the capability to act as a slave terminal but the MediaSynchroniser object is not a slave MediaSynchroniser then this shall be zero. If the terminal does not have the capability to act as a slave terminal then the value of this property shall be undefined. If the MediaSynchroniser object is a slave MediaSynchroniser then the initial value shall be Number.POSITIVE_INFINITY and shall then be updated at between 250 ms and 1 000 ms intervals with the limit on synchronization accuracy achievable, measured in milliseconds, as estimated by the WC-Client function of the terminal (see clause 13.7.4) in the period since the last time this property was updated. When this property is updated, an InterDeviceSyncDispersionUpdate event shall be triggered.

function onInterDeviceSyncDispersionUpdate()	
Description	The function that is called when the interDeviceSyncDispersion property of a MediaSynchroniser object of a slave terminal has been updated.

readonly Number minSyncBufferSize	
Description	The size in bytes of the buffer for synchronization as described by the buffer model in clause 13.5 if implemented, else zero. This value is the minimum guaranteed size of buffer available and is not expected to change during the lifecycle of an HbbTV® application.

readonly Number maxBroadbandStreamsWithBroadcast	
Description	The number of broadband streams the terminal supports for multi-stream synchronization if one stream in the multi-stream synchronization is a broadcast.

readonly Number maxBroadbandStreamsNoBroadcast	
Description	The number of broadband streams the terminal supports for multi-stream synchronization if there is no broadcast stream in the multi-stream synchronization.

readonly Number currentTime	
Description	The value of this property shall be the current playback position of the master media, as defined in clause 13.11.3, if • the MediaSynchroniser was initialized using the initMediaSynchroniser() method or • if the MediaSynchroniser was initialized using the initSlaveMediaSynchroniser() method and – the terminal is currently acting as a slave terminal and – at least one Control Timestamp has been received from the master terminal where the most recent had a contentTime that was not null and – the value of the interDeviceSyncDispersion property of the MediaSynchroniser is less than positive infinity. In all other situations, the value of this property shall be NaN. When the value is not NaN, it shall be expressed in units of seconds and fractions of a second and shall meet the precision requirements defined in clause 13.11.3.

8.2.3.2.2 Methods

`void initMediaSynchroniser() (Object mediaObject, String timelineSelector)`		
Description	Initializes a `MediaSynchroniser` for multi-stream synchronization and for inter-device synchronization as a master terminal. After this method has been called, it is only possible to use this `MediaSynchroniser` object as a master for multi-stream synchronization and/or inter-device synchronization. The media object becomes the master media (see clause 13.2.4) and the timeline selected for it defines the timeline used for the `MediaSynchroniser` API when performing multi-stream synchronization and when acting as a master terminal for inter-device synchronization as explained in clause 13.4.3. The media object specified as the parameter to this method shall be automatically added to this `MediaSynchroniser` and therefore cannot subsequently be explicitly added using the `addMediaObject()` method. If the `MediaSynchroniser` has already been initialized (including if it is in a permanent error state) then this call shall fail and an error event shall be triggered. If the media stream for the media object is determined to be not available or if the selected timeline is determined to be not available then this shall result in a permanent error of the `MediaSynchroniser` and an error event shall be triggered. If this method completes without error then the `MediaSynchroniser` shall be considered initialized. When this `MediaSynchroniser` is initialized, if there is an existing `MediaSynchroniser` that has already been initialized then this shall result in a permanent error of the existing `MediaSynchroniser` and it shall trigger an error event.	
Arguments	`mediaObject`	The media object (video/broadcast object, A/V Control object, or an HTML5 media object) that carries the timeline that will be used by the `MediaSynchroniser` API.
	`timelineSelector`	Type and location of the timeline to be used by the `MediaSynchroniser` API. Please refer to clause 13.4.
NOTE: The availability of a timeline can sometimes only be determined some time after the terminal has begun presentation of the stream. This error can therefore occur some time after the MediaSynchoniser is initialized. See clause 9.7.3.		

`void initSlaveMediaSynchroniser (String css ci service url)`		
Description	Initializes a slave `MediaSynchroniser` for inter-device synchronization of the presentation of media objects on this terminal (referred to as the slave terminal) and the media presentation on another terminal (referred to as the master terminal). After this method has been called, it is only possible to use this `MediaSynchroniser` object as a slave for multi-stream synchronization and/or inter-device synchronization. The timeline used for the `MediaSynchroniser` API will be the timeline used for the `MediaSynchroniser` API at the master terminal (see clause 13.4.3). The terminal does not become a slave terminal until the `enableInterDeviceSync()` method is called. If the service endpoint at the specified URL is not available then this shall result in a permanent error of the `MediaSynchroniser` and an error event shall be triggered (see clause 13.3.8). If the `MediaSynchroniser` has already been initialized (including if it is in a permanent error state) then this call shall fail and an error event shall be triggered. If the terminal does not support the capability to act as a slave terminal, then this method shall be undefined. If this method completes without error then the `MediaSynchroniser` shall be considered initialized. When this `MediaSynchroniser` is initialized, if there is an existing `MediaSynchroniser` that has already been initialized then this shall result in a permanent error of the existing `MediaSynchroniser` and it shall trigger an error event.	
Arguments	`css_ci_service_url`	The URL of a DVB CSS CI endpoint at the master terminal. The URL can be retrieved by the discovery API defined in clause 8.2.6.

```
void addMediaObject (Object               mediaObject,
                     String               timelineSelector,
                     CorrelationTimestamp correlationTimestamp,
                     Number               tolerance,
                     Boolean              multiDecoderMode)
```

<table>
<tr><td>Description</td><td colspan="2">Adds a media object, i.e. video/broadcast object, A/V Control object or HTML5 media object, to the MediaSynchroniser. If the <code>MediaSynchroniser</code> was initialized with the <code>initMediaSynchroniser()</code> method, or if inter-device synchronization has been enabled, then the terminal shall start to synchronize the media object to other media objects associated to this <code>MediaSynchroniser</code> as a result of this method call. The behaviour of the media objects when this method is called is defined in clause 9.7.1.

If the <code>MediaSynchroniser</code> is not initialized, or is in a permanent error state, then this call shall be ignored and an error event dispatched.

If the media object has already been added to the <code>MediaSynchroniser</code>, then this call shall be ignored and an error event dispatched.

If adding the media object would result in multi-stream synchronisation using a combination of streams that is unsupported by the terminal, then this call shall be ignored and a transient error of the <code>MediaSynchroniser</code> shall be generated with error code 20.

The actual presentation of the content might be delayed while the terminal aligns the master media object and the other media object(s) to achieve synchronized presentation in accordance with the correlation timestamps.

The terminal may be required to buffer one or more of the media objects. If the terminal has insufficient buffer space or cannot present the media sufficiently early then the media object shall be added to the <code>MediaSynchroniser</code> but a transient error of the <code>MediaSynchroniser</code> shall be generated with error code 1 or 11.

The terminal shall select the components from the media object to be presented in accordance with the value of the <code>multiDecoderMode</code> parameter and the definitions in clause 10.2.7.

If the terminal fails to access a media item or its timeline, e.g. the resource is not available, then adding the media object shall fail and the <code>MediaSynchroniser</code> shall dispatch an error event.

If the correlation timestamp <code>correlationTimestamp</code> is undefined a correlation timestamp where the value of both properties is 0 shall be assumed. If the correlation timestamp is null or has an invalid format, adding the media object shall fail and the terminal dispatch an error event.

If both of the following conditions apply:
• The correlation timestamp <code>correlationTimestamp</code> was passed as <code>undefined</code> to this method; and
• the type of the timeline used for the master media is PTS and the type of the timeline for this media object is also PTS;
then the rules stated in clause 10.2.8.2 shall apply. In all other cases the rules stated in clause 10.2.8.3 shall apply.</td></tr>
<tr><td rowspan="4">Arguments</td><td><code>mediaObject</code></td><td>video/broadcast object, A/V Control object, or an HTML5 media object</td></tr>
<tr><td><code>timelineSelector</code></td><td>Type and location of the timeline used for the media object's timeline. Please refer to clause 13.4.</td></tr>
<tr><td><code>correlationTimestamp</code></td><td>An optional initial correlation timestamp that relates the media objects timeline to the synchronization timeline.</td></tr>
<tr><td><code>tolerance</code></td><td>An optional synchronization tolerance in milliseconds. The tolerance, if provided, is expressed as a positive value including 0. If the application passes a negative value or does not provide this argument then the terminal shall assume a tolerance of 0.

See clause 9.7.2 for details on how a terminal uses this value.

This argument does not define the accuracy of synchronization to be achieved by the terminal, such as frame accuracy or lip-sync accurate.</td></tr>
</table>

	multiDecoderMode	An optional parameter that defines whether component selection for this media object is performed separately (as defined in clause 10.2.7.3) or collectively with other media objects on this MediaSynchroniser (as defined in clause 10.2.7.4). If the value does not equal true, the terminal shall follow the rules for component selection in clause 10.2.7.3. If the value is true and the terminal does not support multiple decoders or currently does not have sufficient resources it shall ignore the call and dispatch an error event with the error code 2. Otherwise the terminal shall follow the rules in clause 10.2.7.4. Applications should check the extraSDVideoDecodes and extraHDVideoDecodes properties before using this parameter set to true.

void removeMediaObject (Object mediaObject)		
Description	Removes an object from this MediaSynchroniser. The terminal shall not stop the presentation of media objects purely as a result of this method call. However, if there are insufficient decoders available to continue to present this media object, the presentation of the media object may cease due to insufficient resources. If the media object has not already been added to the MediaSynchroniser or is the master media object then this call shall be ignored and an error event dispatched. If the MediaSynchroniser is not initialized, or is in a permanent error state, then this call shall be ignored and an error event dispatched.	
Arguments	mediaObject	The media object to be removed.

`void updateCorrelationTimestamp (Object mediaObject, CorrelationTimestamp correlationTimestamp)`		
Description	Updates the correlation timestamp for the media object, which relates the media object's timeline to the timeline used by the `MediaSynchroniser` API. A correlation timestamp should be updated as often as necessary if the correlation of the timelines changes, e.g. due to drift or discontinuity event of one timeline. If the correlation timestamp is null or has an invalid format then this call shall be ignored and an error shall be dispatched with error code 5. When an updated correlation timestamp is set for a media object which would change the temporal relative presentation of the media objects, the terminal shall adapt to this. Example mechanisms for achieving this gracefully can be found in clause C.3 of TS 103 286-2 [47]. If the change to the correlation timestamp would cause the terminal to run out of buffer space or be unable to present the media sufficiently early then an error shall be dispatched with error code 1 or 11 as appropriate. If the change to the correlation timestamp would cause the terminal to be able to achieve synchronized presentation of this media object when previously it could not, then an `onSyncNowAchievable` event shall be generated for the `MediaSynchroniser` object (see clause 8.2.3.2.3). If the media object is not already added to the `MediaSynchroniser` then this call shall be ignored and an error event dispatched. If the `MediaSynchroniser` is not initialized, or is in a permanent error state, then this call shall be ignored and an error event dispatched.	
Arguments	`mediaObject`	The media object for which the correlation timestamp shall be set.
	`correlationTimestamp`	The correlation timestamp to be used with this media object.

`void enableInterDeviceSync (function callback)`		
Description	Enables inter device synchronization of a master terminal or slave terminal. If it is already enabled then this call shall be ignored. If the `MediaSynchroniser` was initialized using the `initMediaSynchroniser()` method then the terminal become a master terminal as defined in clause 13.3.3. If the `MediaSynchroniser` was initialized using the `initSlaveMediaSynchroniser()` method then the terminal become a slave terminal as defined in clause 13.3.5. The callback method shall be called when the endpoints are operable. The `nrOfSlaves` property can be used to poll for the number of connected slave terminals or companion applications. If the `MediaSynchroniser` is not initialized, or is in a permanent error state, then this call shall be ignored and an error event dispatched.	
Arguments	`callback`	Optional callback function.

void disableInterDeviceSync (function callback)		
Description	Disables the inter device synchronization of a master or slave terminal. If the terminal is a master terminal it shall cease to be a master terminal as defined in clause 13.3.4. Once the terminal is no longer a master terminal then the callback function shall be called. If the terminal is a slave terminal it shall cease to be a slave terminal as defined in clause 13.3.6. Once the terminal is no longer a slave terminal then the callback function shall be called. If the MediaSynchroniser is initialized but the terminal is currently neither a master nor a slave terminal then the callback function shall be immediately called. If the MediaSynchroniser is not initialized, or is in a permanent error state, then this call shall be ignored and an error event dispatched (see clause 13.3.8).	
Arguments	callback	Optional callback function.

8.2.3.2.3 DOM2 events

When an error occurs for the MediaSynchroniser object then the intrinsic event onError shall be generated and a corresponding DOM level 2 event shall be generated in the following manner:

Intrinsic event	Corresponding DOM 2 event	DOM 2 Event properties
onError	Error	Bubbles: No Cancelable: No Context Info: lastError, lastErrorSource

For the intrinsic event onSyncNowAchievable , a corresponding DOM level 2 event shall be generated, in the following manner:

Intrinsic event	Corresponding DOM 2 event	DOM 2 Event properties
onSyncNowAchievable	SyncNowAchievable	Bubbles: No Cancelable: No Context Info: mediaObject

For terminals with the capability to act as a slave terminal, then for the intrinsic event "onInterDeviceSyncDispersionUpdate", a corresponding DOM level 2 event shall be generated, in the following manner:

Intrinsic event	Corresponding DOM 2 event	DOM 2 Event properties
onInterDeviceSyncDispersionUpdate	InterDeviceSyncDispersionUpdate	Bubbles: No Cancelable: No Context Info: None

NOTE: These DOM 2 events are directly dispatched to the event target, and will not bubble nor capture. Applications SHOULD NOT rely on receiving these events during the bubbling or the capturing phase. Applications that use DOM 2 event handlers have to call the addEventListener() method on the MediaSynchroniser embedded object. The third parameter of addEventListener(), i.e. "useCapture", will be ignored.

8.2.3.2.4 Error codes

The values of the error property are defined in this clause.

Value	Description	Permanent or Transient
1	Synchronization is unachievable because the terminal could not delay presentation of content (represented by a media object added using the `addMediaObject()` method) sufficiently to synchronize it with the master media. For example: the buffer size for media synchronization is not sufficient.	Transient
2	The presentation of media object(that was added using the `addMediaObject()` method) failed. The specific reason is given by the error handler of that media object.	Transient
3	The media or the selected timeline for the media could not be found or the media timeline is no longer present (for media represented by a media object that was added using the `addMediaObject()` method).	Transient
4	Media object is already associated with the `MediaSynchroniser`.	Transient
5	The correlation timestamp set for a media object is `null` or has an invalid format.	Transient
6	Inter-device synchronization with a master terminal failed because of unavailability, e.g. an endpoint is not available or disappeared. Applications should rediscover available terminals as defined in clause 14.7.2 before continuing with inter-device synchronization.	Permanent
7	The call failed because the `MediaSynchroniser` is not yet initialized.	Transient
8	The media object referenced as an argument in the call needed to have already been added to the `MediaSynchroniser` using the `addMediaObject()` method, but it has not been.	Transient
9	The media object (that was passed using the `addMediaObject()` method) is not in a suitable state to participate in synchronization. See clause 9.7.1.	Transient
10	Inter-device synchronization with a master terminal failed because of a fault in protocol interaction, e.g. the master terminal did not provide required messages or data. Applications can consider trying again.	Permanent
11	Synchronization is unachievable because the terminal could not present the content (represented by a media object added using the `addMediaObject()` method) sufficiently early to synchronize it with the master media.	Transient
13	The method call failed because the `MediaSynchroniser` is in a permanent error state or because it has been replaced by a newer initialized `MediaSynchroniser`.	Transient (see note 1)
14	The presentation of the master media (that was specified as an argument when the `initMediaSynchroniser()` method was called) failed. The specific reason is given by the error handler of that media object.	Permanent
15	The master media object or the selected timeline for a media object could not be found or the media timeline is no longer present.	Permanent
16	The master media object is not in a suitable state to participate in synchronization. See clause 9.7.1.	Permanent
17	The method call failed because the `MediaSynchroniser` is already initialized.	Transient
18	The `MediaSynchroniser` has been replaced by a new `MediaSynchroniser` being initialized.	Permanent
19	The master terminal has reported that the `presentationStatus` of the master media has changed to "transitioning" (see clause 13.6.3).	Transient
20	The combination of streams requested for multi-stream synchronisation (by a call to the `addMediaObject()` method) is unsupported.	Transient
NOTE: The `MediaSynchroniser` will already be in a permanent error state. If this error occurs, the `MediaSynchroniser` remains in the permanent error state.		

Behaviour of the `MediaSynchroniser` object depending on whether the error is transient or permanent is defined in clauses 13.3.7 and 13.3.8.

8.2.3.3 The CorrelationTimestamp class

8.2.3.3.1 General

Applications shall construct objects that conform to this class definition to deliver correlation timestamps to the terminal.

8.2.3.3.2 Properties

`Number tlvMaster`	
Description	A value on the timeline used by the `MediaSynchroniser` API.

`Number tlvOther`

Description	A value on the timeline of a media object that correlates to the value on the timeline used by the `MediaSynchroniser` API.

The significance of the properties of this class for both multi-stream and inter-device synchronization are discussed in more detail in clause 13.4.

8.2.4 APIs for automatic deletion of downloaded content

The following additional property on the `Download` object (as defined in the OIPF DAE specification [1]) shall be supported.

`Boolean flaggedForDeletion`	
Description	Boolean property set by the application indicating whether the content item has been flagged as suitable for automatic deletion. When additional storage space is needed to complete a download that has been registered, if deleting content items flagged for automatic deletion would provide the additional storage space needed then enough of these content items shall be deleted by the terminal to provide the space needed. When a download is removed automatically as a result of this flag being set, the effect shall be the same as the effect of the `remove()` method: the download and any data and media content associated with it shall be removed, and all properties on the `Download` object shall be set to undefined. The storage space corresponding to the downloads which are flagged for deletion shall be deemed as free by the `checkDownloadPossible()` method. The order in which items flagged for deletion are actually removed by the terminal, and the time at which they are removed, is implementation-dependent, and may take into account factors such as the time at which the item was flagged for deletion, the size of the item compared to the amount of space needed. Items flagged for deletion shall not be removed automatically if they are currently being referred to by an A/V Control object or a `<video>` element.

8.2.5 APIs for obtaining the LCN of a service

The following additional property on the `Channel` class (as defined in OIPF DAE specification [1]) shall be supported.

`readonly Number terminalChannel`	
Description	An integer property which shall be set to the value of the terminal's Logical Channel Number as used by the terminal's native UI. This allows for terminals to have different channel values (for example by way of user sorting) from the LCN values provided in SI by the broadcast network.

The property `majorChannel` (as defined in OIPF DAE specification [1]) from the `Channel` class shall be supported with the following definition.

`readonly Number majorChannel`	
Description	An integer property that shall be set to the value of the Logical Channel Number as defined by the logical channel information provided by the SI from the broadcast network. The values shall be equal to the final LCN assignments after any sorting performed by the terminal as part of the channel scan only, taking into account other SI descriptors relating to network sorting (such as `logical_channel_descriptor`, `HD_Simulcast_Logical_Channel_descriptor`, `target_region_descriptor`, etc., as defined by the country profile). It shall also be unaffected by any re-assignment of the LCN due to the terminal retaining and re-numbering duplicate services.

8.2.6 Companion Screen discovery APIs

8.2.6.1 HbbTVCSManager embedded object

This embedded object shall have the MIME type "`application/hbbtvCSManager`". It enables applications to:

- discover companion screens with a running Launcher application and for launching / installing CS applications;
- discover the base URLs of the local and remote endpoints for application to application communication;
- discover other HbbTV® terminals on the home network;
- discover the URLs of service endpoints provided by other HbbTV® terminals on the home network;
- launch or install a CS application on a Companion Screen.

In this clause, the definitions of connected and disconnected are as defined in clause 14.3.

Before a CS application can be launched / installed via the Launcher application, connected Launcher applications have to be discovered by the HbbTV® application. This can be achieved by using the `discoverCSLaunchers()` API:

`Boolean discoverCSLaunchers(function onCSDiscovery)`		
Description	Triggers a callback reporting CS launcher applications on the home network, along with their enumeration ID, a friendly name and their CS OS information. This returns with either the value `true` to indicate that the function has completed with no errors (and that a callback is expected), `false` otherwise. When `true` is returned, the `onCSDiscovery()` callback shall be scheduled to fire within 1 second. There shall be no callback scheduled if `false` is returned. The details of what is done during this function call or after this function call dependes on the protocol between the HbbTV® terminal and the CS launcher application and is implementation specific.	
Arguments	`onCSDiscovery`	A callback function. See below for the details.

The `onCSDiscovery` callback shall be supported and called once for each call to `discoverCSLaunchers()` that returns `true`:

`function onCSDiscovery( Array csLaunchers )`		
Properties	`csLaunchers`	A JavaScript `Array` object containing zero or more `DiscoveredCSLauncher` objects (see clause 8.2.6.3) where each object in the array represents a CS Launcher application that was either: • currently connected at the time of the call to `discoverCSLaunchers()` that triggered this callback; • or subsequently connected after the time of the call to `discoverCSLaunchers()` that triggered this callback. The protocol for determining the CS Launchers to be included in this array is out of scope, and not defined by the present document.
NOTE: There is no mechanism for an HbbTV® application to determine if a CS Launcher application has disconnected from the HbbTV® terminal. The application should make further calls to `discoverCSLaunchers()` as required to keep up to date.		

Boolean discoverTerminals(function onTerminalDiscovery)		
Description	Triggers a callback reporting other HbbTV® terminals on the home network, along with an enumeration ID, a friendly name, and service endpoint URLs. This returns with either the value `true` to indicate that the function has completed with no errors (and that a callback is expected), or `false` otherwise. On HbbTV® terminals that do not support discovering other HbbTV® terminals, `false` shall be returned. When `true` is returned, the `onTerminalDiscovery()` callback shall be scheduled to fire within 1 second. There shall be no callback scheduled if `false` is returned.	
Arguments	`onTerminalDiscovery`	A callback function. See below for the details.

The `onTerminalDiscovery` callback shall be supported and called once for each call to `discoverTerminals()` that returns `true`:

function onTerminalDiscovery (Array terminals)		
Arguments	`terminals`	A JavaScript `Array` object containing zero or more `DiscoveredTerminal` objects (see clause 8.2.6.2) where each object in the array represents an HbbTV® terminal that was either: • available for connecting to the HbbTV® terminal (at the time of the call to `discoverTerminals()`); • or subsequently became available for connecting to the HbbTV® terminal (i.e. after the call to `discoverTerminals()`. Terminals shall use the protocol defined in clause 14.7 to discover other terminals for inclusion in this array.
NOTE: There is no mechanism for an HbbTV® application to determine if another HbbTV® terminal is no longer available. The application should make further calls to `discoverTerminals()` as required to keep up to date.		

To launch or install a CS application on a Companion Screen the `launchCSApp()` method needs to be called for the Companion Screen identified by the Companion Screen enumeration ID (`enum_id`). The action that the CS Launcher application on the Companion Screen will undertake is described by the payload string. The semantics of the instruction and the payload format are described in clause 14.4.2.

Boolean launchCSApp(Number enum_id, String payload, function onCSLaunch)		
Description	Sends a payload string to the CS Launcher application which contains an instruction set for the CS Launcher application to execute. The result of the Launch operation is communicated to the HbbTV® application via the `onCSLaunch` callback. The protocol for achieving this is out of scope, and not defined by the present document. The function returns false if the `enum_id` does not refer to a launcher application, otherwise it returns true.	
Arguments	`enum_id`	The unique ID of an instance of a CS Launcher application.
	`Payload`	See clause 14.4.2 for the definition of format of the payload parameter string.
	`onCSLaunch`	A callback function. See below for the details.

When the result of the launch operation is known, the HbbTV® browser calls the `onCSLaunch` callback:

callback onCSLaunch(int enum_id, int error_code)		
Properties	`enum_id`	A unique ID for a CS Launcher application
	`error_code`	See below for the error codes

The following error codes may be carried in the `onCSLaunch` callback:

Error Code	Numeric value	Error Description
`op_rejected`	0	The CS Launcher application has automatically rejected the operation with no interaction with the user of the Companion Screen. NOTE: This error code is intended for use when the CS Launcher has automatically blocked the operation due to a blacklist feature.
`op_denied`	1	The CS Launcher application has blocked the operation, but it was blocked by the explicit interaction of the user of the Companion Screen.
`op_not_guaranteed`	2	The CS Launcher application has initiated the instruction (launch or install) without a problem. It is **assumed** (to the best knowledge of the Launcher application) that the launch or installation operation has completed successfully.
`invalid_id`	3	The CS Launcher application that is identified by `enum_id` is no longer available. (i.e. it has become unavailable since discovery occurred), or has never been available.
`general_error`	4	A general error has occurred that meant that the operation could not be completed for a reason other than those described for the other error codes in this table.
`op_failed`	5	The CS Launcher application attempted a Launch or Install operation but it is known to have failed. For a Native application, it means that the application could not be launched or installed. For an HTML application it means that an HTML application environment could not be launched.
`app_already_installed`	6	The CS Launcher application received an Install operation (without a Launch operation) but it detected that the requested application is already installed
NOTE: The above error codes names are **not** available as pre-defined properties anywhere. A HbbTV® application has to either use the integer values or define the error names itself.		

NOTE: CS Launcher apps cannot always detect if a launch or install operation does not complete. An application that receives `op_not_guaranteed` via the callback is recommended to use alternative means to seek confirmation if required. For example: an HbbTV application and launched CS application can use app2app communication to attempt to establish contact with each other.

Since there are certain actions that the Launcher application may undertake before responding (and thus the terminal invoking the `onCSLaunch` callback), there may be a long delay. Applications will therefore be responsible for timing out.

If a CS application needs to use one of the service endpoints defined in clause 14.7.2 (excluding the application Launch service endpoint), then it needs to either:

- be passed these parameters upon launch (as a URL query parameters on the application launch URL); or
- to use the discovery methods defined in clause 14.7.

For the first method to be possible, the calling HbbTV® application needs to be able to determine the locations of the service endpoints so it can construct the launch URL before initiating the launch.

The methods below enable an HbbTV® application to determine the locations of these service endpoints.

`String getInterDevSyncURL()`	
Description	Returns the URL of the CSS-CII service endpoint for the terminal that the calling HbbTV® application is running on. The URL retrieved by this method shall be the same as the URL carried in the `<X_HbbTV_InterDevSyncURL>` element as described in clause 14.7.2.

`String getAppLaunchURL()`	
Description	Returns the URL of the application launch service endpoint for the terminal that the calling HbbTV® application is running on. The URL retrieved by this method shall be the DIAL application Resource URL for HbbTV® (for this terminal) as defined in clause 14.7.2.

`String getApp2AppLocalBaseURL()`	
Description	Returns the base URL of the application to application communication service local endpoint, as defined in clause 14.5.2. The use of this endpoint to communicate between the HbbTV® application and the remote client is described in clause 14.5.1. The URL retrieved by this method shall end with a slash ('`/`') character.

`String getApp2AppRemoteBaseURL()`	
Description	Returns the base URL of the application to application communication service remote endpoint. The URL retrieved by this method shall be the same as the URL carried in the `<X_HbbTV_App2AppURL>` element as described in clause 14.7.2 and shall end with a slash ('`/`') character.

8.2.6.2 DiscoveredTerminal class

Instances of this class provide details of endpoints of a Terminal that has been discovered using the `discoverTerminals()` method of the `HbbTVCSManager` object (see clause 8.2.6.1).

A `DiscoveredTerminal` object shall have the following properties:

`readonly Number enum id`	
Description	A unique ID for a discovered HbbTV® terminal. The `enum_id` is expected to be quasi-static, and that repeated calls to `discoverTerminals()` will respond with the same `enum_id` unless either this terminal or the other terminal have been restarted or the other terminal has been re-connected. Newly started and connected terminals shall generate new `enum_ids`.

`readonly String friendly name`	
Description	A discovered terminal may provide a friendly name, e.g. "`Muttleys TV`", for an HbbTV® application to make use of. It is optional that this parameter is returned. If it is not returned, it shall be set to the empty string "". If set the value shall be carried in the `friendlyName` field of the UPnP device description as required by clause 14.7, and as per the implementation note in clause 5.4 of DIAL [50].

`readonly String X HbbTV App2AppURL`	
Description	The remote service endpoint on the discovered HbbTV® terminal for application to application communication, equal to the value of the element with the same name as defined in clause 14.7.2.

`readonly String X_HbbTV_InterDevSyncURL`	
Description	The remote service endpoint on the discovered HbbTV® terminal for inter-device synchronization, equal to the value of the element with the same name as defined in clause 14.7.2.

readonly String X_HbbTV_UserAgent	
Description	The User Agent string of the discovered HbbTV® terminal, equal to the value of the element with the same name as defined in clause 14.7.2.

8.2.6.3 DiscoveredCSLauncher class

Instances of this class provide details of a discovered CS Launcher that was discovered using the `discoverCSLaunchers()` method of the `HbbTVCSManager` object (see clause 8.2.6.1).

A `DiscoveredCSLauncher` object shall have the following properties:

readonly Number enum_id	
Description	The unique ID of an instance of a CS Launcher application. The `enum_id` is expected to be quasi-static. Repeated calls to `discoverCSLaunchers()` shall respond with the same `enum_id` unless any of the HbbTV® terminal, the Companion Screen, or the CS Launcher application have been restarted or re-connected. Newly started and connected Launcher applications on Companion Screens shall generate new `enum_ids`.

readonly String friendly_name	
Description	A CS Launcher application may provide a friendly name, e.g. "`Muttleys Tablet`", for an HbbTV® application to make use of. It is optional that this parameter is returned. If it is not returned, it shall be set to the empty string "".

readonly String CS_OS_id	
Description	The CS OS identifier string, as described in clause 14.4.1.

9 System integration

9.1 Mapping from APIs to protocols

9.1.1 Unicast streaming

9.1.1.1 General streaming requirements

In Unicast streaming:

- Pausing playback shall cause the video to freeze and the audio to suspend.
- Stopping playback shall cause the video and audio to stop.
- When not presenting video, the A/V Control object shall be rendered as an opaque black rectangle.

NOTE: An A/V Control object that is not presenting video can obscure other parts of the application UI, including video being presented by other elements in the application or in the background.

9.1.1.2 HTTP streaming

The mapping from the APIs for unicast streaming to the protocols shall be as defined in clause 8.2.5.1 of the OIPF DAE specification [1] for HTTP streaming.

9.1.2 Unicast content download

Where unicast content download is supported, the mapping from the APIs for unicast content download to the protocols shall be as defined in clause 8.2.1 of the OIPF DAE specification [1].

9.1.3 Seek accuracy

The play position of media content being presented by a video/broadcast object, an HTML5 media element or an A/V Control object can be controlled using the appropriate seek API.

The information available to the terminal to assist with navigating to a requested seek point depends on the container format and protocol being used. In order to ensure that terminals can always perform a seek with reasonable speed, the following accuracy requirements are defined.

Table 11a – Requirements on seek accuracy

Protocol and system format	Accuracy requirements	Comments
MPEG DASH / ISO BMFF	Seeks to a position shall be performed precisely if: 1) the position is within a live Period and is identifiable from the MPD as being the start of a media segment, or 2) the position is within an on-demand Period and is identifiable from the Segment Index as being the start of a subsegment Seeks to other positions *should* position the media object precisely at the requested position. If they do not, the new media position *shall* be at the nearest position for which precise seeking is required that moves the media position in the direction implied by the seek request. For the definitions of live and on-demand Periods see TS 103 285 [45] clause 4.2.	Seeking accurately to a position that is not the start of a segment or subsegment will typically require the terminal firstly to identify the preceding 'sync sample' in the media segment (see clause 8.6.2 of ISO/IEC 14496-12:2012 [31]) and then to decode but not display the frames from there leading up to the requested position.
HTTP streaming / ISO BMFF	Seeks to a position that is identified as a 'sync sample' (see clause 8.6.2 of ISO/IEC 14496-12:2012 [31]) shall be performed precisely. Seeks to other positions *should* position the media object precisely at the requested position. If they do not, the new media position *shall* be at the nearest identifiable sync sample that moves the media position in the direction implied by the seek request.	Seeking accurately to a position that is not a 'sync sample' will typically require the terminal firstly to identify the preceding 'sync sample' in the media and then to decode but not display the frames from there leading up to the requested position.
HTTP streaming / MPEG-2 TS	Seeks shall preserve the seek direction and shall result in a position error no greater than one GOP length plus 20% of the interval between the last position and the requested position, provided that the media meets the following requirements: • the bitrate averaged over any individual GOP is within ± 10% of the bitrate averaged over the entire media asset • there are no PCR discontinuities anywhere in the media asset	This requirement is intended to permit the terminal to determine byte positions for seek points based on a reasonable measurement of the average stream bitrate, and then to commence playback at the next I-frame.
Broadcast timeshift / MPEG-2 TS	Seeks shall preserve the seek direction and shall position the media object to within 5 seconds or one GOP length of the requested position, whichever is the greater.	The highly variable bitrate of typical broadcast services is likely to require indexing of the timeshift buffer to achieve this.

In all cases, the position on the media timeline reported by the appropriate APIs shall meet the requirements specified for those APIs and shall reflect the true media position. This may mean that the position reported following a seek is different to the position requested in the seek call.

NOTE: The present document does not require support for the fastSeek method defined in HTML 5.1 [51], only the ability to write to the currentTime attribute defined in HTML 5 [54]. This clause takes precedence over requirements defined in those specifications that seeking by writing to the currentTime attribute is frame accurate. The requirements described here are similar to what is described in HTML 5 when the approximate-for-speed flag is set but are more precise in order to be testable.

9.2 URLs

The `http:` and `https:` URL schemes shall be supported as defined in clause 8.3 of the OIPF DAE specification [1], except that support for `https:` is not required for unicast streaming.

It shall be possible to use FDP URLs to refer to files broadcast via FDP, as defined in clause H.2.4.

The `dvb:` URL scheme as defined in TS 102 851 [10] shall be supported and extended as follows:

- It shall be possible to use `dvb:` URLs including path references to refer to DSM-CC file objects and to DSM-CC stream event objects signalled in the current service. It shall be possible to append to URLs referring to DSM-CC file objects an optional query component or fragment component, e.g. to pass parameters to an application. Since '?' and '#' are reserved characters as defined in IETF RFC 3986 [27], if the name of a DSM-CC file object that is part of an HbbTV® application contains such characters, they shall be percent-encoded (as defined in IETF RFC 3986 [27]) when used in URLs.

- It shall be possible to use `dvb:` URLs referring to applications signalled in the current service as defined in Table 4 of TS 102 851 [10] and optionally appended fragment component with the `Application.createApplication()` method. Use of `dvb:` URLs referring to applications from another service will cause `createApplication()` to fail as if the initial page could not be loaded. Any query component and fragment component assigned to this `dvb:`URL shall be attached to the application location URL signalled inside the corresponding AIT as follows:

  - If only one URL contains a query component then the resulting URL shall use that query component.

  - If both URLs contain a query component then the query component of the DVB application URL is appended to the application location URL using an ampersand sign '&'. The terminal shall not parse or process the query components.

  - If only one URL contains a fragment component then the resulting URL shall use that fragment component.

  - If both URLs contain a fragment component, the fragment component of the DVB application URL takes precedence and overwrites the one in the application location URL.

  - The `window.location.href` property shall take the value of the resulting URL, including any query component. Any fragment component shall be available in the `window.location.hash` property and the query component in the `window.location.search` property.

  - Examples for a resulting URL include:

    - URL signaled in the AIT: `http://www.example.com/app1?param1=value1`
      createApplication URL: `dvb://current.ait/1.1?param2=value2#foo`
      Resulting URL: `http://www.example.com/app1?param1=value1&param2=value2#foo`

    - URL signaled in the AIT: `http://www.example.com/app1?param1=value1#test`
      createApplication URL: `dvb://current.ait/1.1#foo`
      Resulting URL: `http://www.example.com/app1?param1=value1#foo`

    - The application is signaled in a DSM-CC Carousel with a Component Tag of 4 and a Base URL of `/index.html?param1=value1` and the current service location is `dvb://1.2.3`
      createApplication URL: `dvb://current.ait/1.1?param2=value2#foo`
      Resulting URL: `dvb://1.2.3.4/index.html?param1=value1&param2=value2#foo`

- Use of `dvb:` URLs referring to files in a carousel carried in a different transport stream shall not cause the terminal to perform a tuning operation, and shall fail as if the file did not exist.

- Use of `dvb:` URLs referring to files in a different carousel carried in the same transport stream shall cause the terminal to unmount the currently mounted carousel and mount the new carousel, as specified in clause 7.2.5.3.

- Support for `dvb:` URLs including the textual service identifier is not required in the present document.

NOTE 1: Some browsers may use the filename suffix as a means for detecting the content type for files (other than HTML documents – see A.2.6.2) not served via HTTP. Application authors should be careful about filename suffixes used, as incorrect suffixes may result in unexpected behaviour.

NOTE 2: The present document inherits requirements to support W3C media fragment URLs from clause 8.3.1 of the OIPF DAE specification [1]. Their use with MPEG DASH content is specified in clause E.4.5 of the present document.

If the HbbTV® terminal supports the CICAM Auxiliary File System resource, it shall support the `ci://` URL scheme:

- An HbbTV® application wishing to access a specific file on a CICAM file system shall use the following URL format to identify the target file:

  - "`ci://`<domain identifier>`/`<file path and name>"

NOTE 3: Domain identifiers except for 'HbbTVEngineProfile1' are outside the scope of the present document. The <file path and name> is private to the application and not defined in the present document.

NOTE 4: CICAMs that want to authenticate the application before enabling access to the CICAM file system may rely on private messaging via `oipfDrmAgent` to only enable the file system to authenticated applications.

NOTE 5: An application that was launched from the CICAM might use the '`HbbTVEngineProfile1`' domain identifier, as specified in clause 11.4.3 of the present document, to access further resources from the CICAM, in addition to any proprietary domain identifiers.

NOTE 6: The requirement to support the `ci://` URL scheme includes the ability to reference a Content Access Streaming Descriptor in combination with the HTML5 video element.

- An XMLHttpRequest to a URL with the `ci://` URL scheme shall be processed by the HbbTV® Browser environment as if it was CORS [42] permitted.

- Access to the content of a file delivered in a carousel shall not be blocked for violating the CORS security policy.

The `data:` URL scheme defined in RFC 2397 [62] shall be supported for inline images in HTML and CSS. Support for `data:` URLs is not required for other uses. Terminals shall support data: URLs of at least 22 000 characters conveying images that are up to 16 384 bytes in size.

9.3 Other file formats

9.3.1 Stream event

Both mechanisms for referencing sources of stream events defined in clause 8.2 of TS 102 809 [3] shall be supported.

For the XML schema defined in clause 8.2 of TS 102 809 [3] the following restrictions shall apply:

- The `stream_event_id` attribute of the type `StreamEventType` shall represent a positive/unsigned integer with a maximum value of 65535. The lexical representation of the value shall be as defined by clause 3.3.23 "unsignedShort" of the W3C XML Schema Recommendation [23].

- The value of the `component_tag` attribute of the type `DsmccObjectType` shall represent a positive/unsigned integer with a maximum value of 255. The lexical representation of the value shall be as defined by clause 3.3.24 "unsignedByte" of the W3C XML Schema Recommendation [23].

- Stream event XML files shall be served with a MIME type of "`application/vnd.dvb.streamevent+xml`".

9.3.2 MPEG DASH event integration

9.3.2.1 General

The DVB DASH specification, TS 103 285 [45] requires support for the event mechanism defined in clause 5.10 of the MPEG DASH specification ISO/IEC 23009-1 [29]. This clause defines how those events will be exposed to applications.

9.3.2.2 HTML5 media element

Specifically the `TextTrackList` shall include `TextTrack`s corresponding to DASH event streams as follows:

- A `TextTrack` shall be provided for each event stream signalled in the MPD as defined in clause 5.10.2 of MPEG DASH ISO/IEC 23009-1 [29] excluding DASH-specific events as defined in clause 5.10.4 of MPEG DASH ISO/IEC 23009-1 [29] and excluding events streams defined by the DVB DASH specification TS 103 285 [45] to be consumed by the terminal.

- A `TextTrack` shall be provided for each event stream included in currently selected Representations as defined in clause 5.10.3 of MPEG DASH ISO/IEC 23009-1 [29] excluding DASH-specific events as defined in clause 5.10.4 of MPEG DASH ISO/IEC 23009-1 [29] and excluding events streams defined by the DVB DASH specification TS 103 285 [45] to be consumed by the terminal.

Changes in the set of event streams, either by updating the MPD or by changing the selected Representation, shall be reported using the `onaddtrack`/`addtrack` and `onremovetrack`/`removetrack` events.

The mapping between DASH event streams and `TextTrack` objects shall be as follows:

TextTrack property	MPD Events	Inband Events
Kind	Metadata	Metadata
Label	Empty string	Empty string
Language	Empty string	Empty string
Id	Empty String	Empty String
inBandMetadataTrackDispatchType	@schemeIdUri + "U+0020" (SPACE character) + @value	@schemeIdUri + "U+0020" (SPACE character) + @value
Mode	Hidden	Hidden

DASH events shall be reported to applications as `DataCues` according to the following mapping:

DataCue(TextTrackCue) property	MPD Events	Inband Events
Id	@id	Id
startTime	@presentationTime (scaled according to the EventStream @timescale attribute) + the time offset of the start of the period from the start of the presentation.	presentation_time_delta (scaled according to the timescale value) + the time offset of the start of the segment from the start of the presentation.
endTime	The startTime + @duration, subject to the minimum duration requirements below. If the @duration attribute is not specified, endTime shall be set to `Number.MAX_VALUE`.	The startTime + the event_duration, subject to the minimum duration requirements below. If event_duration is 0xFFFF, endTime shall be set to `Number.MAX_VALUE`.
pauseOnExit	False	False
Onenter	As defined in the HTML5 Recommendation [54].	As defined in the HTML5 Recommendation [54].
Onexit	As defined in the HTML5 Recommendation [54].	As defined in the HTML5 Recommendation [54].
data	The string value of the `<Event>` element.	`message_data`

The `cuechange` event of the `TextTrack` object shall be fired according to the "time marches on" algorithm defined in clause 4.7.10.8 of the HTML5 Recommendation [54]. This allows the possibility of "missed cues" (cues that start and end between successive iterations of the algorithm). For these cues a cuechange event will be fired but the cue will not be available in the `activeCues` `TextTrackCueList` when the handler is called.

To ensure that it is possible to deliver event data to an application using the DASH event mechanism in a way that does not lead to "missed cues", the following requirements shall be observed:

- For any DASH event with a duration of 250 ms or less, the terminal shall set the `endTime` property of the corresponding `DataCue` object to the `startTime` + 250 ms.
- For any `DataCue` with a duration of at least 250 ms, the Terminal shall ensure that a `cuechange` event is raised with the cue listed in the `activeCues` list.

NOTE: The figure of 250 ms is consistent with the minimum repetition rate required by the HTML5 Recommendation [54] for the `timeupdate` event.

Terminals should attempt to minimize the delay from a DASH event occurring and the cuechange event being fired. In circumstances under which no user events or application controlled XHR requests are being processed, and where DASH events occur no more frequently than four times a second, terminals should fire the `cuechange` event within 80 ms of the time a video frame with the event's `presentationTime` would be composed on screen and combined with the application's graphics. Lower levels of accuracy can be expected to degrade the viewer experience where application behaviour is synchronized with A/V media.

Some events may be intended for consumption by the DASH client itself. These are described in clause 10.1.4 of the DVB DASH specification TS 103 285 [45]. With the exception of these events the `cues` attribute of the `TextTrack` shall be populated as follows:

- For an MPD `EventStream`, the `cues` attribute shall contain cues representing the complete list of DASH Events currently defined for that `EventStream`. If the MPD is dynamic, the list shall be updated if the list of Events changes following an MPD update.
- For an `InbandEventStream`, the `cues` attribute shall contain cues representing at least the DASH Events whose start times are less than or equal to the current playback position and whose end times are greater than the current playback position. In addition, past cues shall be retained in the cue list at least until the completion of the next iteration of "time marches on" that occurs after the end time of the cue. The cue list may also contain additional past or future Events which the terminal has acquired.

9.4 Presentation of adaptive bitrate content

9.4.1 General

Terminals shall support the `<ContentURL>` element of the content access streaming descriptor referencing an MPD as defined in DASH ISO/IEC 23009-1 [29].

It is optional for a terminal to support play speeds other than 0 or 1 for adaptive bitrate content.

If paused, terminals shall not auto-resume if DASH live media no longer exposes the current play position via its time shift buffer (as determined by the `MPD@timeShiftBufferDepth`). The current playback position shall be maintained unless the application requests a change, or there is an error.

9.4.2 Behaviour for HTML5 media objects

Media timeline

The origin of the media timeline used by HTML5 media elements (i.e. the `<audio>` and `<video>` elements) shall be the start time of the first Period that was defined in the MPD when the MPD was first loaded. The origin of the media timeline shall not change unless the HTML5 media element source is changed or the `load()` method is called.

NOTE 1: Implementations are expected to be able to handle past periods being removed from a dynamic MPD without changing the origin of the HTML5 media element's timeline.

For a dynamic MPD, `getStartDate()` shall return a `Date` object representing the value of `MPD@availabilityStartTime` plus the PeriodStart time (see clause 5.3.2.1 of MPEG DASH ISO/IEC 23009-1 [29]) of the first regular Period when the MPD was first loaded.

NOTE 2: This provides an absolute time reference for the start of the media timeline used by the HTML5 media element.

For a static MPD or a dynamic MPD containing no regular Period, `getStartDate()` shall return a `Date` object that represents the time value `NaN`, in the same manner as required by HTML5 when no explicit date and time is available.

Duration

For a static MPD, the `duration` attribute shall be the value of `MPD@mediaPresentationDuration` if present or the `PeriodStart` time of the last Period determined according to clause 5.3.2.1 of MPEG DASH ISO/IEC 23009-1 [29] plus the value of `Period@duration` for the last Period. The duration shall be calculated after resolution of any xlink references with `@xlink:actuate` set to "`onLoad`".

For a dynamic MPD, the `duration` attribute shall be the value of `MPD@mediaPresentationDuration` if present, otherwise it shall be reported as positive infinity (indicating an indeterminate duration).

Seekable range

When a dynamic MPD contains an `MPD@timeShiftBufferDepth` attribute, the media element's `seekable` attribute shall take values that map to the full range of media available in the server-side time shift buffer that the terminal could present, taking into account any safety margins (see TS 103 285 [45], clause 4.7). The range shall be calculated based on segment availablility times as defined in clause 5.3.9.5.3 of ISO/IEC 23009-1 [29]. The range shall be the intersection of the ranges derived from all selected AdaptationSets.

NOTE 3: The duration of the seekable range will be at most the value of `MPD@timeShiftBufferDepth`. If the terminal has a time uncertainty of +/- T seconds, this would typically have the effect of increasing the start and decreasing the end of the range by T seconds. If the terminal has a minimum time behind the live edge for presentation of a live stream, this will further reduce the end of the seekable range.

For a static MPD, or where no `MPD@timeShiftBufferDepth` attribute is present in a dynamic MPD, the `seekable` attribute shall reflect the full extent of the media timeline currently defined by the MPD.

NOTE 4: For a dynamic MPD, this range may not begin at time zero if Periods have been removed since the MPD was first retrieved.

Pause and Resume behaviour

After a live DASH stream is paused, if the current play position (the `currentTime` attribute) is no longer in the time shift buffer (as determined by the `MPD@timeShiftBufferDepth`) when the video playback is attempted to be resumed, then an error Event with code `MEDIA_ERR_NETWORK` shall be raised. The application can then determine the new seekable range and act accordingly.

Seeking

If the current play position is modified (by changing the `currentTime` attribute), and that new position is outside the seekable range defined above, the terminal shall follow the seek behaviour defined by HTML5.

Change in size of the time shift buffer

If playing, and the current play position (the `currentTime` attribute) is no longer within the time shift buffer (for example if there was a dynamic MPD update that shortens the `MPD@timeShiftBufferDepth`), then an error Event with code `MEDIA_ERR_NETWORK` shall be raised.

Start Position

The start position shall be determined according to the requirements laid out in TS 103 285 [45], clause 10.9.2. For some content, playback will start at the 'live edge'. This is consistent with the requirements of HTML5 [54] which includes the following step when processing media data:

> "If either the media resource or the address of the current media resource indicate a particular start time, then set the initial playback position to that time"

9.4.3 Behaviour for the A/V Control object

Terminals shall support applications setting the `data` attribute of an A/V Control object to a URL referencing an MPD as defined in DASH ISO/IEC 23009-1 [29] and identified by the MIME type in annex C of [29]. The `type` attribute of the A/V Control object shall be set to "`application/dash+xml`".

In order to play the content, the terminal shall fetch the MPD from the URL, interpret the MPD and select an initial set of representations. If at any time the MPD is found to be not valid according to the XML schema or semantics defined in DASH ISO/IEC 23009-1 [29], the A/V Control object shall go to play state 6 ('error') with error value 4 ('content corrupt or invalid').

When an instance of the `AVComponent` class refers to a DASH audio media content component:

- If the audio media component is identified as being audio description (as defined in clause E.2.4), the `audioDescription` property of the `AVComponent` shall be set to true.

The origin of the media timeline used for the `playPosition` property shall be the start time of the first Period that was defined in the MPD when the MPD was first loaded. The origin of the media timeline shall not change unless the A/V Control object returns to the stopped state.

NOTE: Implementations are expected to be able to handle past periods being removed from a dynamic MPD without changing the origin of the timeline used for the `playPosition` property.

For a static MPD, the `playTime` property shall be the value of `MPD@mediaPresentationDuration` if present or the PeriodStart time of the last Period determined according to clause 5.3.2.1 of MPEG DASH ISO/IEC 23009-1 [29] plus the value of `Period@duration` for the last Period. The duration shall be calculated after resolution of any xlink references with `@xlink:actuate` set to "`onLoad`".

For a dynamic MPD, the `playTime` property shall be the value of `MPD@mediaPresentationDuration` if present, otherwise it shall be reported as positive infinity (indicating an indeterminate duration).

Seekable range

Since the A/V Control object does not provide an API or attribute to determine the seekable range, an application is responsible for determining the seekable range by other means if it required.

Pause and Resume

After a live DASH stream is paused, if the current play position (the `playPosition` property) is no longer in the DASH sliding window when the video playback is attempted to be resumed, then a playback error has occurred and the A/V Control object transitions to the play state 6 ('error') with a detailed error code of 6 ('content not available at given position').

Since the A/V Control object does not provide an API to determine the seekable range, an application could recover playback by stopping the media and then restarting the media.

Seeking

If the application attempts to seek a live DASH stream outside the range of the time shift buffer (as determined by the `MPD@timeShiftBufferDepth`), then the seek request is rejected and the current playout is maintained as per clause 7.14.1.1, clarification 7, of the OIPF DAE specification [1].

Change in size of the time shift buffer

If playing, and the current play position (the `playPosition` property) is no longer within the time shift buffer (for example if there was a dynamic MPD update that shortens the `MPD@timeShiftBufferDepth`), then a playback error has occurred and the A/V Control object transitions to the play state 6 ('error') with a detailed error code of 6 ('content not available at given position').

9.5 Downloading content via FDP

9.5.1 Download registration

Download of content delivered via FDP can be registered by using the `registerDownload()` or `registerDownloadURL()` methods, as defined in the OIPF DAE specification [1].

To do so, these methods are called with a Content Access Download Descriptor (or a URL of a Content Access Download Descriptor) including the content items to be downloaded via FDP. Since the FDP mechanism relies on the signaling of availability windows indicating when the content is being transmitted (as specified in clause H.3.2), these methods shall fail (as defined in the OIPF DAE specification [1] and amended in clause A.2.19 of the present

document) if no availability window is specified for a content item to be downloaded using FDP, i.e. under any of the following circumstances:

- The URL argument passed to `registerDownloadURL()` is an FDP URL, as defined in clause H.2.4.
- At least one of the `<ContentURL>` elements in a `<contentItem>` element included in the Content Access Download Descriptor is an FDP URL, while this `<contentItem>` element does not include at least one `<availabilityWindow>` element.

Moreover, when a `<ContentURL>` element in the Content Access Download Descriptor is an FDP URL, the `transferType` element shall be assumed to have the value `full_download`, regardless of its signalled value.

9.5.2 Single file with multiple URLs

When one Content Access Download Descriptor includes multiple `<ContentItem>` elements containing FDP URLs pointing to the same file (according to the criteria defined in clause H.2.3), the terminal shall consider that all these `<ContentItem>` elements correspond to one single content item, i.e. that the corresponding file shall be downloaded only once. In such a case, the terminal may assume that these `<ContentItem>` elements only differ by their `<ContentURL>` and `<availabilityWindow>` elements.

NOTE: This occurs in cases where one single file is broadcast several times at different times, resulting in different availability windows, and possibly at different locations (e.g. on different transport streams), resulting in different FDP URLs (see clause H.3.3 for details).

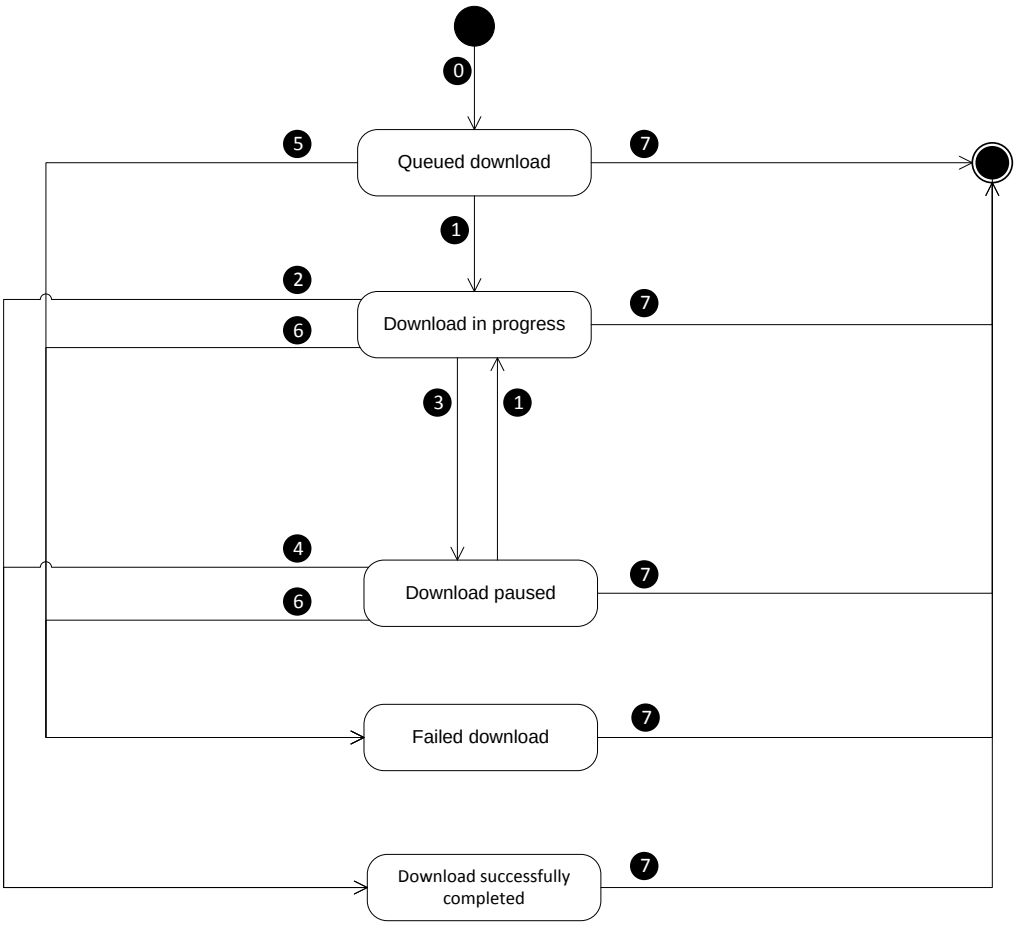
9.5.3 Properties of the Download object

In the case of a download via FDP, the following properties of the `Download` object shall be constrained in the following manner:

1) `timeElapsed`: this element shall have the value `undefined`.
2) `timeRemaining`: this element shall have the value `undefined`.
3) `currentBitRate`: this element shall have the value `undefined`.
4) `startTime`: this element shall have the value `undefined`.
5) `totalSize`:
    a. Before an availability window has started and a first FDP Initialization Message has been received, the `totalSize` property shall correspond to the value of the `size` attribute of the selected `<ContentURL>` element in the Content Access Download Descriptor that was used to register the download.
    b. Once an availability window has started and a first FDP Initialization Message has been received, the `totalSize` property shall be updated to the actual file size as indicated in the `file_size` field of the FDP Initialization Message.
    c. If the download reaches the state "failed", its `totalSize` property shall be set to 0.
6) `state` and `suspendedByTerminal`:
    a. The `state` property shall change from "queued" to "in progress" at the receipt of the first Initialization Message.
    b. If a download has not yet completed, and there are availability windows in the future, but there is no active availability window at present, then `state` shall be "paused", and `suspendedByTerminal` shall be true.
    c. When the last availability window of a download has ended, `state` shall be either "failed" or "completed".

9.5.4 Download state diagram

In the case of a download via FDP, the state diagram of section 7.4.3.1. of the OIPF DAE specification [1] does not apply, and the diagram shown in figure Figure 16 below shall prevail.



0 registerDownload() or registerDownloadURL() is called, with valid arguments.

1 An availability window has started and at least one FDP Initialization message has been received.

2 (a) The file has been entirely downloaded, or
(b) After the end of the last known availability window, the terminal successfully retrieved the missing or erroneous segments thanks to the File Segment Recovery mechanism, or
(c) discard_file_on_error_flag = 0 and the end of the last known availability window is passed, and File Segment Recovery has been performed (succesfully or not), or is not attempted.

3 The end of an availability window has been reached while the file is not entirely downloaded, and there is at least one availability window known to occur in the future.

4 (a) After the end of the last known availability window, the terminal successfully retrieved the missing or erroneous segments thanks to the File Segment Recovery mechanism, or
(b) discard_file_on_error_flag = 0 and the end of the last known availability window is passed, and File Segment Recovery has been performed (succesfully or not), or is not attempted.

5 (a) The end of the last known availability window is passed, or
(b) The Content Access Download Descriptor fetched at the URL supplied as argument of registerDownloadURL() is not accepted.

6 discard_file_on_error_flag = 1 and the end of the last known availability window is passed while the file is not entirely downloaded, and the terminal did not successfully retrieve (or did not attempt to retrieve) the missing or erroneous segments thanks to the File Segment Recovery mechanism.

7 remove() has been called, or the download has been automatically deleted by the terminal as a result of the flaggedForDeletion boolean property being set.

Figure 16: State diagram for embedded application/oipfDownloadManager objects for FDP downloads (normative)

9.6 Media element integration

9.6.1 General

The media elements from HTML5 [54] are included in the present document through the reference to the OIPF DAE specification [1]. Terminals shall support presenting content using the media elements as follows:

- Non-adaptively streamed video and/or audio as defined in clause 7.3.2.1 of the present document shall be supported with the content identified by an HTTP URL either directly referring to the content or referring to a content access streaming descriptor that in turn refers to the content.
- Adaptively streamed video and/or audio as defined in annex E of the present document shall be supported identified by an HTTP URL either directly referring to an MPD or referring to a content access streaming descriptor that in turn refers to the MPD.
- If the download option is supported then presenting downloaded content shall be supported with that content identified by a URL returned by the `uri` property of the `Download` class.
- If the PVR option is supported then presenting recorded content shall be supported with that content identified by a URL returned by the `uri` property of the `Recording` class.

9.6.2 Resource management

The terminal shall support the existence within the same DOM of at least one HTML5 media element that is playing together with at least two HTML5 media elements in a paused state, where each HTML5 media element may be in a `readyState` of `HAVE_CURRENT_DATA` or higher. The terminal shall support each of the following scenarios;

- all three HTML5 media elements refer to DASH content
- all three HTML5 media elements refer to ISOBMFF content
- two of the three HTML5 media elements refer to DASH content and one refers to ISOBMFF content
- two of the three HTML5 media elements refer to ISOBMFF content and one refers to DASH content

The terminal may use hardware audio and video decoders to decode and render `<video>` and `<audio>` HTML5 media elements. These hardware resources shall not be allocated to an HTML5 media element before it changes from being paused to 'potentially playing' (as defined in the HTML5 specification). When subsequently paused, an HTML5 media element shall retain its hardware resources, but shall be able to release these resources if required to start playing another HTML5 media element. When resources are released, the terminal may discard any decoded frames that have not been displayed.

The visual appearance of a `<video>` element that has no decoder resource currently allocated is undefined. It is recommended that the terminal render such an element using the same behaviour as if the "`visibility`" CSS property was equal to "`hidden`".

If a terminal supports only one HTML5 media element that is 'potentially playing', and multiple media elements exist within the DOM, the request to transition to 'potentially playing' of one HTML5 media element (e.g. calling the `play()` method) shall cause all other media elements to pause and release their allocated hardware resources. The transition to 'potentially playing' shall be deferred until all other HTML5 media elements have paused and released their hardware resources. HTML5 media elements that are forced to pause shall emit a "pause" event and set the "`paused`" attribute to true.

NOTE: The policy for managing hardware resources between instances of the HTML5 media element defined here (automatically releasing allocated hardware resources when a new request occurs) is intentionally the exact opposite of the policy defined for the A/V control and video/broadcast objects by the OIPF DAE specification [1] and refined by clause A.2.1 of the present document.

See clause A.2.1 of the present document for the policy for managing hardware resources between instances of the HTML5 media element and instances of the A/V control or video/broadcast objects.

Buffering of video in an HTML5 media element shall be possible while broadcast video is presented in a video broadcast object or by the terminal.

9.6.3 Transition behaviour

The delay between the end of presentation of an HTML5 media element and starting presentation of another HTML5 media element shall be less than 250 ms if all of the following conditions are met:

- the `pause()` function on the first HTML5 media element and the `play()` function on the second HTML5 media element are called within the same spin of the event loop;
- the `readyState` of the second HTML5 media element has reached `HAVE_FUTURE_DATA` or greater (as indicated by the "canplay" event);
- the start position in the media stream of the second video element is a random access point. In the case of H.264 video, a random access point is an instantaneous decoding refresh (IDR) access unit; for AAC audio only streams, a random access point is the start of any access unit.
- the first media element refers to either MPEG DASH content or ISOBMFF content and the second media element refers to either MPEG DASH content or ISOBMFF content

If the `readyState` is less than `HAVE_FUTURE_DATA`, there may be additional unbounded delay as audio/video data is downloaded.

If the start position does not coincide with an IDR, there may be additional delay as the hardware decoder decodes the frames between the IDR and the start position.

When resuming the playback of an HTML5 media element that has previously been paused, the terminal shall start playback at or before the IDR following the pause position. When resuming the playback, the terminal should start playback as close as possible to the pause position, preferably from the next frame following the pause position.

9.6.4 Reporting and control of buffering

The terminal shall provide information on the amount of data that is buffered for each HTML5 media element by providing a TimeRanges object via the `buffered` attribute of the HTML5 media element. The accuracy of the range (or ranges) in the `TimeRanges` object shall be within ±T seconds of the actual amount of media that has been buffered by the terminal, where T is:

- the segment duration, when playing fragmented content (such as DASH);
- 5 seconds, otherwise.

The terminal should support control of the media buffering by implementing the `preload` attribute of the HTML5 media element. When the `preload` attribute is set to "`none`", the terminal should not download audio/video content for this media element. When rendering a media format that uses manifest or playlist files (such as DASH, MSS and HLS) the terminal may continue to download these files when the `preload` attribute is set to "`none`". When the `preload` attribute is set to "`metadata`" the terminal should download audio/video content for this media element at a reduced rate, for example by downloading slower than real-time. When the `preload` attribute is set to "`auto`" the terminal may choose any downloading strategy, including using as much bandwidth as is available.

9.6.5 Distinguishing multiple media tracks (informative)

When content items include multiple video tracks and/or audio tracks, applications may wish to use a wide range of properties to decide which track should be presented at a particular time or under particular conditions. The HTML5 `VideoTrack` and `AudioTrack` interfaces permit video tracks and audio tracks to be distinguished by 3 properties - `id`, `language` and `kind` (e.g. subtitles, translation, commentary, description, captions, etc.). They have no explicit support for distinguishing video tracks and audio tracks based on other properties such as codec, DRM, video aspect ratio and number of audio channels (stereo or 5.1 or 7.1). Some examples of tracks that cannot be explicitly distinguished include:

- Stereo and multi-channel (5.1/7.1) versions of the same audio.
- HE-AAC and Dolby versions of the same audio.
- AVC and HEVC versions of the same video.

The above is equally true for multiple video and/or audio Adaptation Sets in an MPEG DASH MPD.

If a content item contains tracks that cannot be explicitly distinguished based on language and kind and if applications wish to control which audio track or which video track is presented under these circumstances then some possible approaches an application may use include the following:

- provide some mechanism to determine the relevant properties from the id of the `VideoTrack` or `AudioTrack` (e.g. making an XMLHttpRequest to a server providing that information); or
- if the system format permits track ids to be assigned by the content provider then it may be practical for the content provider to use a convention when assigning the ids that permits additional track properties to be determined by an application;
- present the content item using the A/V Control object instead of the HTML5 `<video>` element. With the A/V Control object, the `AVComponent` class and its sub-classes provide properties exposing this information.

9.6.6 Controls attribute

Applications should not set the `controls` attribute on the video media elements. Terminal behaviour if this is set is implementation dependent.

NOTE: In clause 4.7.10.13 "User interface", the HTML5 specification as referenced by [i.6] states that "Even when the attribute is absent, however, user agents may provide controls to affect playback of the media resource (e.g. play, pause, seeking, and volume controls), but such features should not interfere with the page's normal rendering." clause 10.2.7 of the present document requires terminals to provide some controls.

9.6.7 DRM

If an application attempts to present DRM protected content using the HTML5 `<video>` element and this is denied by the DRM system then this failure shall be reported to the application by a `MediaError` whose `code` property is set to `MEDIA_ERR_DECODE`. The application is then responsible for checking if the reason for this error was related to DRM and if so, obtaining more details about the error from the DRM system. For DRM systems that an application can access through the `oipfDrmAgent` object, these two steps would be done using the `sendDrmMessage` method.

Subject to rights being available, all features defined to work with unencrypted video shall also work with encrypted video except as follows;

- Limitations apply to CICAM player mode due to the feature set of the protocol between the HbbTV terminal and the CICAM. For more details, see clauses 10.2.8.1, 10.2.9.1 and K.5.
- The present document does not require terminals to support decrypting two streams at the same time. Hence in the scenario described in clause 9.6.2 above, buffering of video in video elements that are not actually playing may be limited to loading encrypted data without it being decrypted.

NOTE: This means that both multi-stream and inter-device media synchronisation (see clause 13) are required to work both with content decrypted using CI Plus host player mode (clause 11.4.2) and embedded DRM (if the DRM feature is supported). Advert insertion is required to work with two encrypted streams; the 250ms performance, however, is required only in cases of one encrypted stream and one unencrypted stream or two unencrypted streams.

9.6.8 Parental Rating Errors

If an application attempts to present content using an HTML5 media element and this is blocked due to parental access control, the application shall receive a `MediaError` with the `code` property set to `MEDIA_ERR_DECODE`.

NOTE: The present document does not provide a way for an application to distinguish failure due to parental access control from failure due to other reasons. If this is important to an application then it may choose to present the content using an A/V Control object instead and listen for a `ParentalRatingChange` event.

9.6.9 Downloaded Content

Clause 7.14.1.3 of the OIPF DAE specification [1] shall apply as follows when an application uses the HTML5 media element to present downloaded content:

- If the download was triggered using `registerDownloadURL()` with a `contentType` other than "`application/vnd.oipf.ContentAccessDownload+xml`" or the download was triggered using a Content Access Download Descriptor with `<TransferType>` value "`playable_download`" as defined in annex E.1 of the OIPF DAE specification [1] and if the `play()` method is called before sufficient data has been download to initiate playback then the HTML5 Recommendation [54] shall be followed as written with the `readyState` set and updated as required in that specification.

- If the downloaded content was triggered using a Content Access Download Descriptor with `<TransferType>` value "`full_download`" as defined in annex E.1 of the OIPF DAE specification [1], and if the `play()` method is called whilst the content is still downloading and has not yet successfully completed, then the method shall fail and a `MediaError` sent with the code set to `MEDIA_ERR_NETWORK`.

9.6.10 Video presentation

Video presented by an HTML5 `<video>` element shall always be presented according to the requirements for full screen mode being false as defined in clause H.2 of the OIPF DAE specification [1] and modified by clause A.2.14 of the present document.

9.6.11 getStartDate method

As defined in clauses A.3.2 and 9.4.2, the `getStartDate()` method shall be supported for MPEG DASH content.

The behaviour of `getStartDate()` for other types of content is not defined by the present document.

9.7 Synchronization

9.7.1 Synchronization and video objects

9.7.1.1 video/broadcast object

A video/broadcast object that is passed to the `initMediaSynchroniser()` or `addMediaObject()` methods shall always be in the connecting or presenting states. Passing a video/broadcast object in any other state to these methods shall result in an error being triggered with error number '9' or '16'. Like timeshift, synchronization shall not impact the state machine of the video/broadcast object. If the video/broadcast object has a permanent error (and hence transitions to the Unrealised state), then:

- If it represents master media (it was passed to a `MediaSynchroniser` via the `initMediaSynchroniser()` method) then this shall result in a permanent error of the `MediaSynchroniser` (see clause 13.3.8) with an error being triggered with error number 14.

- If it represents other media (it was added to a `MediaSynchroniser` via the `addMediaObject()` method) then this shall result in a transient error of the `MediaSynchroniser` (see clause 13.3.7) and the object shall be removed as if an application had called the `removeMediaObject()` method and an error event triggered with error number 2.

If the video/broadcast object transitions to the stopped or unrealised states for any other reason, then:

- If it represents the master media then this shall result in a permanent error of the `MediaSynchroniser` with error code 16.

- If it represents other media then this shall result in a transient error of the `MediaSynchroniser` with error code 9 and the object shall be removed as if an application had called the `removeMediaObject()` method.

The terminal is not required to maintain synchronization of all media objects attached to the `MediaSynchroniser` (including media on other terminals if inter-device synchronization is being performed) to a video/broadcast object that represents the master media (it was passed to a `MediaSynchroniser` via the `initMediaSynchroniser()` method) while one or more of the following conditions are true for that video/broadcast object:

- It has a transient error (that it has not yet recovered from).

- It is in the connecting state.

- It is playing at a speed that is not 0 (paused) or 1 (normal play speed) for reasons other than adjusting its presentation timing to enable media objects representing other media to synchronize to it.

Synchronization of all media objects shall resume automatically when the video/broadcast object representing the master media returns to a state where all the conditions listed above are no longer true (and it has recovered from any transient error).

The terminal is not required to maintain synchronization of a video/broadcast object (that was added to the `MediaSynchroniser` using the `addMediaObject()` method) to the media object representing the master media while that video/broadcast object has a transient error (that it has not yet recovered from) or is in the connecting state.

Synchronization of the video/broadcast object to the media object representing the master media shall resume automatically when the video/broadcast object returns to the presenting state (and hence recovers from any transient error). If the video/broadcast object leaves the connecting state for any other reason than the requirements concerning permanent errors above shall apply. If the `setChannel()`, `prevChannel()`, `nextChannel()`, `pause()`, `resume()`, `setSpeed()`, `seek()` or `stopTimeshift()` methods are called then the method call shall proceed for the video/broadcast object.

Additionally:

- If it represents the master media then the terminal shall adjust the presentation timing of the other media to try to synchronize it with the new presentation timing of the master media.

- If it represents other media then a transient error of the `MediaSynchroniser` with error code 9 shall be generated and the object shall be removed as if an application had called the `removeMediaObject()` method.

9.7.1.2 HTML5 media element

An HTML5 media element that is passed to the `initMediaSynchroniser()` or `addMediaObject()` methods shall have the `readyState` property set to `HAVE_CURRENT_DATA`, `HAVE_FUTURE_DATA` or `HAVE_ENOUGH_DATA`. The media element shall be playing or paused. The `controller` property of the media element shall be `null` or `undefined`. Passing in a media element in any other state to these methods shall result in an error being dispatched with error number 9 or 16. Synchronization may result in the media element being paused, resuming from paused or the currentTime jumping. These shall be reflected in the API as if the application had called the `pause()` method, the `play()` method or written to the `currentTime` property respectively.

NOTE: In the single decoder model (see clause 10.2.7.3), in order to prepare a media object to be passed to the `addMediaObject()` method, an application should use the `load()` method to get the media element into `HAVE_CURRENT_DATA`. Calling the `play()` method will likely fail if another media object that was previously passed to the `initMediaSynchroniser()` method has the hardware video and audio decoders. Even if it does not fail, it may result in a poor user experience if component selection by the terminal is in effect (see clause 10.2.7.2).

If an error occurs while fetching the media data (and hence an error event is triggered), then:

- If it represents master media (it was passed to a `MediaSynchroniser` via the `initMediaSynchroniser()` method) then this shall result in a permanent error of the `MediaSynchroniser` (see clause 13.3.8) with an error being dispatched with error number 14.

- If it represents other media (it was added to a `MediaSynchroniser` via the `addMediaObject()` method) then this shall result in a transient error of the `MediaSynchroniser` (see clause 13.3.7) and the object shall be removed as if an application had called the `removeMediaObject()` method and an error event dispatched with error 2.

If the HTML5 media element source is reloaded (by the application calling the `load()` method), or the application sets an HTML5 `MediaController` for this media element, then:

- If it represents the master media then this shall result in a permanent error of the `MediaSynchroniser` with error code 16.

- If it represents the other media then this shall result in a transient error of the `MediaSynchroniser` with error code 9 and the object shall be removed as if an application had called the `removeMediaObject()` method.

The terminal is not required to maintain synchronization of all media objects attached to the `MediaSynchroniser` (including media on other terminals if inter-device synchronization is being performed) to an HTML5 media element representing the master media (it was passed to a `MediaSynchroniser` via the `initMediaSynchroniser()` method) while one or more of the following conditions are true for that HTML5 media element:

- It is stalled while trying to fetch media data.
- It is playing (not paused) at an "effective playback rate" (as defined in the HTML5 specification) that is not 0 or 1 for reasons other than adjusting its presentation timing to enable media objects representing other media to synchronize to it.

Synchronization of all media objects shall resume automatically when the HTML5 media element representing the master media returns to a state where all the conditions listed above are no longer true.

The terminal is not required to maintain synchronization of an HTML5 media element (that was added to the `MediaSynchroniser` using the `addMediaObject()` method) to the media object representing the master media while that HTML5 media element is stalled while trying to fetch media data.

Synchronization of the HTML5 media element to the media object representing the master media shall resume automatically when sufficient data becomes available.

If the `currentTime` or `playbackRate` properties are set or the `play()` or `pause()` methods are called then the property setting or method call shall proceed for the HTML5 media element. Additionally:

- If it represents the master media then the terminal shall adjust the presentation timing of the other media to try to synchronize it with the new presentation timing of the master media.
- If it represents other media then a transient error of the `MediaSynchroniser` with error code 9 shall be generated and the object shall be removed as if an application had called the `removeMediaObject()` method.

9.7.1.3 A/V Control object

An A/V Control object that is passed to the `initMediaSynchroniser()` or `addMediaObject()` methods shall be in the connecting, buffering, paused or playing states. Passing an A/V Control object in any other state to these methods shall result in an error being dispatched with error number 9 or 16.

NOTE: In the single decoder model (see clause 10.2.7.3), in order to prepare an A/V Control object to be passed to the `addMediaObject()` method, an application should call `play(0)` to get the A/V Control object into the paused state. Calling the `play()` method with a speed other than 0 will likely fail if another media object that was previously passed to the `initMediaSynchroniser()` method has the hardware video and audio decoders. Even if it does not fail, it may result in a poor user experience if component selection by the terminal is in effect (see clause 10.2.7.2).

Synchronization may result in the object transitioning to either the paused or the buffering states and back to the playing states. Such state transitions shall be reported to any application that has registered to receive play state change events for the object concerned. It may result in changes in `playPosition` which shall be reported to any application that has registered to receive `playPositionChange` events. If the A/V Control object transitions to the error state, or loses the connection and transitions to the connecting state, then:

- If it represents master media (it was passed to a `MediaSynchroniser` via the `initMediaSynchroniser()` method) then this shall result in a permanent error of the `MediaSynchroniser` (see clause 13.3.8) with an error being dispatched with error number 14.
- If it represents other media (it was added to a `MediaSynchroniser` via the `addMediaObject()` method) then this shall result in a transient error of the `MediaSynchroniser` (see clause 13.3.7) and the object shall be removed as if an application had called the `removeMediaObject()` method and an error event dispatched with error 2.

If the A/V Control object transitions to the finished or stopped states, then:

- If it represents the master media then this shall result in a permanent error of the `MediaSynchroniser` with error code 16.

- If it represents other media then this shall results in a transient error of the `MediaSynchroniser` with error code 9 and the object shall be removed as if an application had called the `removeMediaObject()` method.

The terminal is not required to maintain synchronization of all media objects attached to the `MediaSynchroniser` (including media on other terminals if inter-device synchronization is being performed) to an A/V Control object representing the master media (it was passed to a `MediaSynchroniser` via the `initMediaSynchroniser()` method) while one or more of the following conditions are true for that A/V Control object:

- It is in the connecting state.
- It is in the buffering state (because insufficient data is available).
- It is in the paused or playing state with a playback speed that is not 0 or 1 for reasons other than adjusting its presentation timing to enable media objects representing other media to synchronize to it.

Synchronization of all media objects shall resume automatically when the A/V Control object representing the master media enters a state where all the conditions listed above are no longer true.

The terminal is not required to maintain synchronization of an A/V Control object (that was added to the `MediaSynchroniser` using the `addMediaObject()` method) to the media object representing the master media while that A/V Control object is in the connecting or buffering states.

Synchronization of the A/V Control object to the media object representing the master media shall resume automatically if the A/V Control object returns to the playing state. If the A/V Control object leaves the connecting or buffering states for any other reason than the object shall be removed from the `MediaSynchroniser` as defined for transitions to the error state above.

If the `data` attribute is set or the `setSource()`, `play()`, `seek()` or `stop()` methods are called then the property setting or method call shall proceed for the A/V Control object. Additionally:

- If it represents the master media then the terminal shall adjust the presentation timing of the other media to try to synchronize it with the new presentation timing of the master media.
- If it represents other media then a transient error of the `MediaSynchroniser` with error code 9 shall be generated and the object shall be removed as if an application had called the `removeMediaObject()` method.

9.7.2 Tolerance

The synchronization tolerance defines the maximum time by which the presentation of two media streams can be out of sync but still considered synchronized. While a video and a corresponding audio stream usually require tight synchronization, an application may choose to loosen this requirement for certain combinations of media streams to allow a terminal to start the synchronized presentation as fast as possible.

The terminal shall interpret the `tolerance` parameter on the `addMediaObject()` method (see clause 8.2.3.2.2) for a media stream as follows. The master media carries the synchronization timeline. The other media is the media stream for which the tolerance value is defined. The synchronization of the other media and master media is deemed within tolerance if the difference in the play position of both streams is not greater than the given tolerance value at any time.

If the media object for the master media is in the playing state (regardless of play speed) when the media object for the other media has been added to a `MediaSynchroniser`:

- If the synchronization of the other media and the master media can be achieved within tolerance without adjusting the timing of the master media stream then the terminal shall not adjust the timing of the master media, but shall instead adjust the timing of the other media such that components of the other media that are to be presented are presented as soon as possible.
- Otherwise if the terminal is a slave terminal and synchronization cannot be achieved then a transient error of the `MediaSynchroniser` object shall be triggered with error code 1 or 11 as appropriate (see clause 8.2.3.2.4).
- Otherwise if synchronization of the other media and the master media can be achieved within tolerance only if the timing of the master media is adjusted and the terminal is able to do so then it shall adjust the timing of the master media such that the components of the other media that are to be presented are presented as soon as possible.

- Otherwise synchronization cannot be achieved and a transient error of the `MediaSynchroniser` shall be triggered with error code 1 or 11 (see clause 8.2.3.2.4).

NOTE: Whether the terminal is able to sufficiently adjust the timing of the master media depends on the type of media stream and whether the terminal supports buffering of the master media and has sufficient buffer space. See clause 13.5.

The terminal shall continue to play the other media and the master media in sync within the synchronization tolerance. The techniques used to do this are implementation dependent but may include changing the playout speed of the other media or the master media to slightly more or slightly less than real time.

If synchronization within tolerance of a media object becomes achievable again after being previously unachievable, then the `MediaSynchroniser` shall generate an `onSyncNowAchievable` event.

For inter-device synchronization, the other media is presented by a slave terminal and the master media is presented by a separate master terminal. The slave terminal shall not take into account inter-device synchronization accuracy as determined by the WC-Client function of the terminal and which is quantified as a measure of dispersion (see clause 13.7.4) when performing the calculations and comparisons described above.

9.7.3 Timeline availability

If at any time the timeline selected by the application for that media stream is determined to be unavailable while the terminal is attempting to synchronize the presentation of a media stream, then:

- If the stream is the master media (it was passed to the `MediaSynchroniser` via the `initMediaSynchroniser()` method) then this shall result in a permanent error of the `MediaSynchroniser` (see clause 13.3.8) and an error shall be dispatched with error number 15.

- If the stream is other media (it was added to the `MediaSynchroniser` via the `addMediaObject()` method) then this shall result in a transient error of the `MediaSynchroniser` (see clause 13.3.7) and an error shall be dispatched with error code 3.

If the requested timeline is an MPEG TEMI timeline (see clause 13.4) then the terminal shall look for the `temi_timeline_descriptor` in the media for 2.5 seconds while the media is in the playing state before determining that the timeline is unavailable and dispatching an error as described above.

NOTE: Broadcasters are recommended to adopt a minimum repetition rate for a `temi_timeline_descriptor` of 1 repetition per second.

If the requested timeline is an MPEG DASH Period Relative timeline (see clause 13.4) then the terminal shall have determined the availability of the timeline once the MPD has been loaded when starting presentation (or updated during presentation) and the `id` attribute of all Periods in the presentation are known.

ISOBMFF Composition Time timelines and MPEG Transport Stream PTS timelines and EBU-TT-D milliseconds timelines are always available while the respective type of media is being presented.

9.7.4 Minimum synchronization accuracy

The minimum accuracy for multi-stream and inter-device synchronization for a given terminal is the largest of:

- 10 ms (being the duration of 1/2 a frame of video at 50fps);

- the duration of 1/2 tick of any timeline selected (during the corresponding `initMediaSynchroniser()` or `addMediaObject()` method call) by the HbbTV® application for any of the streams under the control of the `MediaSynchroniser` object;

- the duration of 1/2 tick of the Synchronization timeline (see clause 13.4.3.2) if inter-device synchronization is being performed.

How this minimum accuracy applies to terminals performing multi-stream synchornisation is defined in clause 10.2.8.1.

How this minimum accuracy applies to terminals while performing inter-device synchronization in the role of a master terminal is defined in clause 13.8.2.4.

How this minimum accuracy applies to terminals while performing inter-device synchronization in the role of a slave terminal is defined in clauses 13.8.3.4 and 13.8.3.5.

9.8 Reliability and resilience

Terminals shall operate reliably in response to rapid user interaction. Specifically, the terminal shall remain fully functional in the following circumstances. Fully functional in this case means at least that the appropriate application at the end of each sequence starts successfully, that it functions as designed and that, where a broadcast service is selected, the video and audio from that service are presented.

- The user changes the selected service 20 times consecutively between two services carrying broadcast-related autostart applications delivered in different carousels, the time interval between requested service changes varying between 50 ms and the greater of (a) the time interval required for the application to start fully and (b) 1 second.
- The user changes the selected service 20 times consecutively between two services carrying broadcast-related autostart applications delivered over broadband, the time interval between requested service changes varying between 50 ms and the greater of (a) the time interval required for the application to start fully and (b) 1 second.
- The user changes the selected service 20 times consecutively between two services carrying broadcast-related autostart applications, one delivered in a carousel, the other delivered over broadband, the time interval between requested service changes varying between 50 ms and the greater of (a) the time interval required for the application to start fully and (b) 1 second.
- A broadcast-related autostart application is terminated manually by the user using the "EXIT or comparable button" mechanism (see clause 6.2.2.11) 20 times consecutively, the time interval between activations of the mechanism varying between 50 ms and the greater of (a) the time interval required for the application to restart fully and (b) 1 second.
- If the terminal supports a means to launch broadcast-independent HbbTV applications from an Internet TV Portal as described in section 5.3.5: a broadcast-independent HbbTV application is started from the Internet TV Portal and then terminated manually by the user 20 times consecutively and then finally started one further time.

Terminals shall be able to present broadcast audio and video reliably when HbbTV applications are launching and stopping. Specifically:

- When the terminal launches a broadcast-related application and broadcast audio or video is being presented and that application does not try to control video playback, there shall not be any artifacts or glitches in presented broadcast audio or video. This includes applications delivered by broadband and DSM-CC object carousel, as well as both autostart and non-autostart applications.
- When the terminal terminates a broadcast-related application and broadcast audio or video is being presented and that application did not try to control video playback, there shall not be any artifacts or glitches in presented broadcast audio or video. This includes where the application calls `destroyApplication()` to exit itself, when application signalling causes the application to be stopped and when it is terminated manually by the user.

Terminals shall be resilient to transient error conditions that are likely to occur, as well as to conditions of low resource availability. Specifically, the terminal shall remain responsive to channel change and application termination requests in the following circumstances:

- The terminal downloads, or attempts to download an XML, HTML, or media file, which has been truncated.
- A broadband application download is prematurely interrupted by a TCP connection reset, or sustained packet loss.
- A broadcast application download is prematurely interrupted by the removal of a carousel from the broadcast stream.
- The terminal attempts to download an initial HTML page with a file size of 100 MB. For the avoidance of doubt, it is not required that such a page be properly loaded and rendered.

- An application attempts to create and initialise an unbounded number of JavaScript arrays, each containing 2 000 000 integers, until the allocation fails.
- The browser raises exceptions that are not explicitly caught by the application.
- An application enters an infinite JavaScript loop including infinite recursion.

10 Capabilities

10.1 Display model

This clause is replaced by annex H, "Display Model" of the OIPF DAE specification [1].

10.2 Terminal capabilities and functions

10.2.1 Minimum terminal capabilities

Minimum terminal capabilities which shall be available to applications are listed in Table 11 for general capabilities. Additional capabilities shall be signalled as defined in clause 10.2.4.

Table 11: Minimum terminal capabilities

	Value	Additional information
HbbTV® application graphic plane resolution	1 280 pixels horizontally by 720 pixels vertically with a 16:9 aspect ratio.	The terminal shall have at least this graphics resolution. The graphics plane may have a higher resolution but the co-ordinate system as seen by applications shall always be 1 280 x 720. Note: this allows for higher resolution rendering of application text and images but limits the granularity with which an application can position graphics. Clause A.3.9 describes APIs that allow applications to exploit the available resolution.
Colour format	RGBA32 should be supported. If an implementation uses lower colour resolutions (e.g. RGBA16) then it shall support at least RGBA4444.	Video overlays supported. Content providers can be expected to author applications with a 32bpp graphics plane in mind. Applications are likely to have a poor visual appearance on terminals that only support a 16bpp graphics plane.

Supported proportional font	"Tiresias™ Screenfont" v8.03 (or equivalent) with the support for the Unicode character range "Generic Western European character set" as defined in annex C of TS 102 809 [3] but excluding the unicode character codes 0149 and 066B.
	The font shall be the default font to be used when none is explicitly specified.
	This font (even if it is an equivalent of "Tiresias™ Screenfont" v8.03) shall be accessible with the following CSS rule:
	`font-family: Tiresias;`
	It shall also possible to use the "`sans-serif`" generic family name to point to the "Tiresias™ Screenfont" v8.03 font (even if other sans-serif fonts are available in the terminal), i.e. "`sans-serif`" shall default to the "Tiresias™ Screenfont" v8.03 font:
	`font-family: sans-serif;`
		Sans-serif, scalable and anti-aliased font.

Supported non-proportional font	"Letter Gothic 12 Pitch" (or equivalent) with the support for the Unicode character range "Generic Western European character set" as defined in annex C of TS 102 809 [3] but excluding the unicode character codes 0149 and 066B.	Monospace, scalable and anti-aliased font.

	Value	Additional information

	This font (even if it is an equivalent of "Letter Gothic 12 Pitch") shall be accessible with the following CSS rule:
	`font-family: "Letter Gothic 12 Pitch";`
	It shall also possible to use the "`monospace`" generic family name to point to the "Letter Gothic 12 Pitch" font (even if other monospace fonts are available in the terminal), i.e. "`monospace`" shall default to the "Letter Gothic 12 Pitch" font:
	`font-family: monospace;`	

Text entry method	Either multi-tap (e.g. as defined in ES 202 130 [i.2]) or an equivalent (e.g. software keyboard) where characters are input character by character in the text field. For multi-tap or other methods which use multiple supported key events to generate a single character, these intermediate key events shall not be reported to applications. Only the final character result shall be reported to applications.	The `input-format` CSS property may be used by terminals to determine which text entry method to use. Multi-tap also known as SMS-tap is not to be confused with T9 text entry which is not required.
Minimum number of DSM-CC related section filters	The terminal shall allocate sufficient resources to acquire DSM-CC sections from at least 3 elementary streams simultaneously for a given DSM-CC carousel. In addition, a terminal shall reserve at least one section filter for monitoring DSM-CC StreamEvent's events.	
Minimum DSM-CC cache size	The terminal shall reserve 3 MByte for caching objects carried in DSM-CC object carousels.	
Minimum File System Acceleration cache capabilities	The terminal shall reserve at least 8 MByte for the FSA cache. Terminals where a hard disc is accessible by HbbTV® shall reserve at least 64 MByte for the FSA cache. The terminal shall support at least 1024 files carried across 64 stored groups (i.e. average 16 files per group) in the FSA cache. Where a hard disc is available the terminal shall support at least 4096 files carried across 256 stored groups (i.e. average 16 files per group) in the FSA cache. The terminal should perform FSA caching of a carousel over one cycle of this carousel (assuming that there are no transport errors The terminal shall support persistence of FSA cache as long as the terminal is operating HbbTV® normally. Assuming that the HbbTV® environment terminates in a controlled manner then the FSA cache contents shall be available to HbbTV® when it next executes. For example: • If the terminal transitions in a controlled manner from operating HbbTV® to another interactive environment (e.g. MHEG or TV portal application) or other content source than broadcast (e.g. DLNA, HDMI etc.) then the FSA contents will be available to HbbTV® next time it runs. • If power to the terminal is unexpectedly lost then the FSA cache contents may not be available to HbbTV® next time it runs.	These cache sizes refer to non-compressed data. The size information in manifest file also applies to file data in non-compressed form. The FSA cache addressed here is independent of the Mininum DSM-CC cache size specified in this table.
System layer for unicast streaming using HTTP and file download	Both MPEG-2 TS and MP4 file format (as defined in clause 7.3.1.2) shall be supported.	
Video formats for unicast streaming using	Both AVC_SD_25 and AVC_HD_25 shall be supported (as defined in clause 7.3.1.3).	

	Value	Additional information
HTTP and file download		
Audio format for unicast streaming using HTTP and file download	HEAAC, E-AC3 and MPEG1_L3 as defined in clauses 7.3.1.1 and 7.3.1.4.	
Audio format for audio from memory	For audio played as defined by clause 7.14.10 of the OIPF DAE specification [1], HEAAC shall be supported (as defined in clause 6.3.2 of the OIPF DAE specification [1]). For audio played by the Web Audio API [65], the following shall apply; • Non-interleaved IEEE 32-bit linear PCM shall be supported as defined in clause 2.9 of Web Audio [65]. • MPEG1_L3 (as defined in clause 8.1 of the OIPF Media Formats specification [2]) shall be supported for the `AudioContext.decodeAudioData()` method.	
PVR management	If the PVR feature is supported, the `manageRecordings` attribute of the recording capability shall have the value "`samedomain`".	See clause 9.3.3 of the OIPF DAE specification [1].
Download management	If content download is supported, the `manageDownloads` attribute of the download capability shall have the value "`samedomain`".	See clause 9.3.4 of the OIPF DAE specification [1].
Simultaneous demultiplexing of broadcast and broadband content	Required (see 6.2.2.6.3).	
Parental rating scheme	Terminal shall at least support the scheme of a minimum recommended age encoded as per EN 300 468 [16].	
Video scaling	Terminals shall be able to present video at sizes down to 1/8 by 1/8 of the width and height of the logical video plane - equivalent to 160 x 90 pixels in the HbbTV® application graphics plane. Terminals shall be able to scale video down to 1/4 by 1/4 and should be able to scale video down to 1/8 by 1/8. For sizes between 1/4 by 1/4 and 1/8 by 1/8, terminals which cannot scale video shall crop the video instead and display it centered in the according video object of the HbbTV® application graphics plane. Terminals shall be able to scale video up to 2 x 2 of the width and height of the logical video plane. Within these limits, any arbitrary scaling factor shall be allowed. The aspect ratio of decoded video shall be preserved at all scaling factors. See OIPF DAE annex H.2 [1] for more information.	
Cookie support	Cookies with an expiry date shall be stored in persistent memory. Terminals shall respect the expiry date of the cookie. Terminal shall follow IETF RFC 6265 [24] when implementing cookies support. Since clause 6.1 of IETF RFC 6265 [24] does not fix strict limits, the present document fix the following minimum capabilities that terminals shall simultaneously support: • At least 4 096 bytes per cookie (as measured by the sum of the length of the cookie's name, value, and attributes). • At least 20 cookies per domain. • At least 100 cookies total.	As implied by IETF RFC 6265 [24], if a cookie or a "`Set-Cookie`" header is bigger than the maximum size supported by the terminal, it will be discarded, not truncated.

	Value	Additional information
	• At least 5 120 bytes for the "`Set-Cookie`" header. Subject to the requirements of the same origin policy, the same store of cookies shall be accessible to an HbbTV application regardless of whether it is broadcast-related or broadcast-independent, and regardless of the mechanism by which the application was launched, such as any of the mechanisms listed in clause 6.2.2.6.2.	
Web Storage	The data stored through the Web Storage API shall be stored in persistent memory. The present document is intentionally silent about the amount of storage available through the Web Storage API.	
Simultaneous WebSocket connections initiated from the terminal	The number of WebSocket connections from the browser or user agent that can be open simultaneously.	Greater than or equal to 20.
FDP download performance	Terminals shall be capable of downloading a minimum of two files simultaneously via FDP, and shall support a cumulated download bit rate of at least 5 Mbit/s, measured over any 100 millisecond time period.	
Browser HTTP cache	Terminals shall allocate at least 3 Mbytes of storage to the browser HTTP cache (see clause 7.3.2.6).	

An equivalent font is one for which all the following are true:

- The line height of both fonts is the same.
- The widths of the glyphs for corresponding character points are the same in both fonts (where the character point is defined in both fonts).
- The kerning tables contain the same values for both fonts where both of the character points in the pair are present in both fonts.
- Either the appearance of the glyphs is visually similar or they are valid glyph variants as defined by unicode.

10.2.2 User input

10.2.2.1 Key events

Implementations shall provide a mechanism for the end user to generate key events as defined in Table 12.

Table 12: Key events and their status

Button (for conventional remote controls)	DOM-2 Key event	Status	Availability
4 colour buttons (red, green, yellow, blue)	`VK_RED, VK_GREEN, VK_YELLOW, VK_BLUE`	Mandatory	Always available to applications
4 arrow buttons (up, down, left, right)	`VK_UP, VK_DOWN, VK_LEFT, VK_RIGHT`	Mandatory	Always available to applications
ENTER or OK button	`VK_ENTER`	Mandatory	Always available to applications
BACK button	`VK_BACK`	Mandatory	Always available to applications
Number keys	`VK_0` to `VK_9` inclusive	Mandatory	Only available to applications once activated
Play, stop, pause	`VK_STOP` and either `VK_PLAY` and `VK_PAUSE` or `VK_PLAY_PAUSE`	Mandatory	Only available to applications once activated
Fast forward and fast rewind	`VK_FAST_FWD` `VK_REWIND`	Mandatory	Only available to applications once activated
Record	`VK_RECORD`	Mandatory if the PVR feature is supported, otherwise optional.	Only available to applications once activated
TEXT or TXT or comparable button	Not available to applications	mandatory	
2 program selection buttons (e.g. P+ and P-)	Not available to applications	Optional	
WEBTV or comparable button	Not available to applications	Optional	
EXIT or comparable button	Not available to applications	Mandatory	

Key events which have a key code listed in Table 12 shall be available to all applications when requested through the `Keyset` object. Key events which do not have a key code listed in Table 12 shall be handled by the implementation and not delivered to applications.

The availability column indicates if the key events are always available to applications or only once the application has been activated. Key events listed as "Only available to applications once activated" shall be available to applications only once the user has activated the application. Applications AUTOSTARTed by the terminal shall be activated when they have received a key event. Other applications (e.g. broadcast-independent applications or ones signalled as PRESENT) shall be activated when launched. The key set of an application shall only contain keys that are available to the application at that time. If a key set is requested that includes keys not available to an application then that part of the request shall be discarded and only any remaining part of the request relating to available keys shall be processed. When an application becomes activated, the key set shall not automatically change, the application needs to call `Keyset.setValue()` in order to receive key events that were not previously available to it but now are. In all cases, applications shall only receive key events when they have focus as defined below.

Applications shall not rely on receiving any key events not requested through a `Keyset` object, for example when the end user is inputting text into an input field. However, the set of key events requested via a `Keyset` object only identifies the minimum set of keys that may be sent to an application, and so applications should not rely on receiving only those key events.

On up, down, left, right keydown events, terminals shall choose one of the following navigation mechanisms in the priority order listed below:

- Allow applications to capture the events and prevent the default action (known as "JavaScript navigation").
- Handle CSS3 directional focus navigation when the `nav-up`, `nav-right`, `nav-down` and `nav-left` CSS properties are used by the application.
- A default navigation mechanism provided by the terminal which shall allow focus to be moved between navigable elements and allow all navigable elements to gain focus.

Applications shall set the `NAVIGATION` bit of the keyset object even if the navigation keys are only used for focus based navigation (including the CSS `nav-*` properties) and not used in JavaScript event handlers.

The context in which an HbbTV® application runs has more than one type of input focus.

- Focus within a document or page as defined in clause 7.4.3 of the HTML5 Recommendation [54].
- Whether the browser or user agent has input focus or whether something else in the terminal has input focus. This is referred to as "system focus" in the HTML5 Recommendation [54].
- If the browser or user agent supports running multiple documents at the same time then only one of these can have input focus at one time. This may be separate from whether the browser or user agent itself has input focus.

When the browser has "system focus" and the document corresponding to the HbbTV® application has input focus within the browser then the HbbTV® application shall have input focus. If either or both of these cease to apply then the HbbTV® application shall lose input focus until either the application terminates or both apply again. If an HbbTV® application loses input focus then a blur event shall be sent to the application's `Window` object. If an HbbTV® application that has lost input focus regains it then a focus event shall be sent to the application's `Window` object.

While an HbbTV® application has input focus, it shall receive all of the key events shown as "mandatory" in Table 12above that it has requested in its keyset. If any other feature of the terminal needs to use any of those key events then diverting any of those key events to that feature shall result in loss of input focus for the HbbTV® application. The HbbTV® application shall lose focus even if the other feature of the terminal concerned only uses, for example, left, right and enter or green.

If the other feature of the terminal only uses those key events temporarily and all the "mandatory" key events in Table 12 become available to the HbbTV® application then this shall result in the HbbTV® application regaining focus. Some examples of such a temporary loss of focus by the HbbTV® application could include conditional access UI dialogues, parental access control UI dialogues and trailer booking UI dialogues.

When the focus is on either i) an `input` element of a type that accepts text input (e.g. type="text" or "search") or ii) a `textarea` element then all key events that can be generated by the "Text entry method" required by Table 11 "Minimum terminal capabilities" (e.g. virtual keyboard) shall be delivered to the element with focus regardless of whether those key events are in application's current KeySet.

10.2.2.2 Mouse and wheel events

If a terminal indicates that it has pointer support in the capability profile as defined in clause 9 of the OIPF DAE specification [1], implementations shall provide a mechanism for the end user to generate mouse events and may provide a mechanism for the end user to generate wheel events as defined in clause 9.3.24 of the OIPF DAE specification [1]. Any mouse or wheel events generated shall be dispatched to applications only once the application has been activated (see clause 10.2.2.1).

In all cases, applications shall only receive mouse and wheel events when they have focus as defined in clause 10.2.2.1.

Applications shall not rely on receiving any mouse or wheel events they have not requested by using the `Keyset.supportsPointer` property in clause 7.2.5.2 of the OIPF DAE specification [1]. However, the set of mouse and wheel events defined by OIPF only identifies the minimum set of mouse and wheel events that may be sent to an application, and so applications should not rely on receiving only those mouse and wheel events.

10.2.3 Terminal functions

10.2.3.1 Favourites and bookmarks

The terminal should provide a feature to organize frequently used broadcast-independent interactive applications as bookmarks or favourites.

For the presentation of applications on manufacturer portals or in favourite lists the terminal may use a title and an icon specified in the HTML head section and the URL of the initial page of the application:

- The application name shall be defined by the HTML title element.
- The server delivering the application may use the HTTP Accept-Language header to adapt any textual information to the preferred language of the user.
- The linking to an application icon shall be done by an HTML `<link>` element with the following attributes. See also the W3C note on adding a Favicon to a site [i.4]:
  - `rel` - shall have the value '`icon`';
  - `type` - shall contain the mime type of the image format;
  - `href` - shall be the URL of the image.
- The image format and mime types of the icon shall be as defined in clause 7.1.1.
- An application may have multiple icons for different aspect ratios, e.g. 4 by 3 and square. It is recommended that an application provides at least one icon with a square aspect ratio.

10.2.3.2 Streaming and Download

Terminals shall not permit persistent storage of broadband delivered content whose delivery was initiated using the streaming APIs (the A/V Control object or an HTML5 media element) whether streamed adaptively using MPEG DASH as defined in annex E or non-adaptively. Service providers who want to offer content for persistent download should use the download API.

Terminals may use persistent storage media to transiently buffer broadband delivered content for the purposes of multi-stream synchronization between the broadband delivered content and broadcast content as described in clause 13.5.

Terminals that use persistent storage media for the purposes of timeshifting or recording broadcasts may use the same persistent storage media for timeshifting or recording broadband delivered content to be played back in synchronization with broadcast delivered content. Broadband delivered content that is stored in this way shall not be permitted to be presented separately from the broadcast content that it was synchronized with.

10.2.3.3 PVR

It is up to the terminal to decide whether PVR feature related calls are executed directly or if additional means to determine whether to allow the call for the application are employed, such as opening a dialog to query the user.

10.2.3.4 Download via broadcast using FDP

When the resources of the terminal (e.g. tuners) are concurrently required for performing a download via FDP and for TV viewing, priority shall always be granted to TV Viewing.

However, the present document is intentionally silent about priorities between download via FDP and PVR recording.

The terminal shall be able to wake up automatically from standby states when an availability window starts for a download via FDP which has been registered and is not completed, in order to perform/complete the corresponding download operation.

NOTE: Power management in general and in particular the definition of a standby state are outside the scope of the present document.

Performance requirements for file download via FDP are given in clause 10.2.1 of the present document.

10.2.4 HbbTV® reported capabilities and option strings

The `xmlCapabilities` property of the `application/oipfCapabilities` embedded object shall describe an XML document that conforms to the schema defined in clause A.2.15.

For a terminal supporting only the base level of features, the XML `Document` object provided by the `xmlCapabilities` property of the `application/oipfCapabilities` embedded object shall describe an XML document that when canonicalized according to the W3C XML Canonicalization specification [28] shall be equal to the canonicalized form of the following XML:

```
<profilelist xmlns="urn:hbbtv:config:oitf:oitfCapabilities:2014-1"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="urn:hbbtv:config:oitf:oitfCapabilities:2014-1 config-hbbtv-
oitfCapabilities.xsd">
  <ui_profile name="OITF_HD_UIPROF+DVB_S+TRICKMODE">
  <ext>
    <parentalcontrol schemes="dvb-si">true</parentalcontrol>
    <clientMetadata type="dvb-si">true</clientMetadata>
    <temporalClipping />
  </ext>
  </ui_profile>
  <audio_profile name="MPEG1_L3" type="audio/mpeg"/>
  <audio_profile name="HEAAC" type="audio/mp4"/>
  <audio_profile name="MP4_HEAAC" type="audio/mp4" transport="dash" sync_tl="dash_pr"/>
  <video_profile name="MP4_AVC_SD_25_HEAAC" type="video/mp4" transport="dash"
                 sync_tl="dash_pr" />
  <video_profile name="MP4_AVC_HD_25_HEAAC" type="video/mp4" transport="dash"
                 sync_tl="dash_pr" />
  <video_profile name="MP4_AVC_SD_25_HEAAC_EBUTTD" type="video/mp4" transport="dash"
                 sync_tl="dash_pr" />
  <video_profile name="MP4_AVC_HD_25_HEAAC_EBUTTD" type="video/mp4" transport="dash"
                 sync_tl="dash_pr" />
 <video_profile name="TS_AVC_SD_25_HEAAC" type="video/mpeg" sync_tl="temi" />
  <video_profile name="TS_AVC_HD_25_HEAAC" type="video/mpeg" sync_tl="temi" />
  <video_profile name="MP4_AVC_SD_25_HEAAC" type="video/mp4" />
  <video_profile name="MP4_AVC_HD_25_HEAAC" type="video/mp4" />
  <html5_media>true</html5_media>
</profilelist>
```

"`DVB_S`" shall be replaced by the appropriate string(s) for the supported broadcast delivery system(s).

Other parental control schemes in addition to "`dvb-si`" may be listed in the `<parentalcontrol>` element.

Only the video format profiles supported for broadband shall be listed.

As mentioned in Table 9, the terminal may also support E-AC3 audio, in which case the following elements shall be added after the elements listed in the `<profilelist>` element in the above XML:

```
<audio_profile name="MP4_E-AC3" type="audio/mp4" />
<audio_profile name="MP4_E-AC3" type="audio/mp4" transport="dash" sync_tl="dash_pr" />
<video_profile name="TS_AVC_SD_25_E-AC3" type="video/mpeg" sync_tl="temi" />
<video_profile name="TS_AVC_HD_25_E-AC3" type="video/mpeg" sync_tl="temi" />
<video_profile name="MP4_AVC_SD_25_E-AC3" type="video/mp4" />
<video_profile name="MP4_AVC_HD_25_E-AC3" type="video/mp4" />
<video_profile name="MP4_AVC_SD_25_E-AC3" type="video/mp4" transport="dash" sync_tl="dash_pr" />
<video_profile name="MP4_AVC_HD_25_E-AC3" type="video/mp4" transport="dash" sync_tl="dash_pr" />
<video_profile name="MP4_AVC_SD_25_E-AC3_EBUTTD" type="video/mp4" transport="dash"
               sync_tl="dash_pr" />
<video_profile name="MP4_AVC_HD_25_E-AC3_EBUTTD" type="video/mp4" transport="dash"
               sync_tl="dash_pr" />
```

Terminals that support HEVC UHD video as defined in clause 7.3.1.3 shall include the following video profiles:

```
<video_profile name="MP4_HEVC_UHD_25_HEAAC_EBUTTD" type="video/mp4" transport="dash"
               sync_tl="dash_pr"/>
<video_profile name="MP4_HEVC_UHD_25_HEAAC_EBUTTD" type="video/mp4" />
```

and, if E-AC3 audio is supported in the broadcast channel, shall additionally include the following video profiles:

```
<video_profile name="MP4_HEVC_UHD_25_E-AC3 EBUTTD" type="video/mp4" transport="dash"
               sync_tl="dash_pr"/>
<video_profile name="MP4_HEVC_UHD_25_E-AC3 EBUTTD" type="video/mp4" />
```

Terminals that support 8-bit HEVC HD video and not 10-bit HEVC HD video as defined in clause 7.3.1.3 shall include the following video profiles:

```
<video_profile name="MP4_HEVC_HD_25_8_HEAAC_EBUTTD" type="video/mp4" transport="dash"
               sync_tl="dash_pr"/>
<video_profile name="MP4_HEVC_HD_25_8_HEAAC_EBUTTD" type="video/mp4" />
```

and, if E-AC3 audio is supported in the broadcast channel, shall additionally include the following video profiles:

```
<video_profile name="MP4_HEVC_HD_25_8_E-AC3_EBUTTD" type="video/mp4" transport="dash"
               sync_tl="dash_pr"/>
<video_profile name="MP4_HEVC_HD_25_8_E-AC3_EBUTTD" type="video/mp4" />
```

Terminals that support 10-bit HEVC HD video as defined in clause 7.3.1.3 shall include the following video profiles:

```
<video_profile name="MP4_HEVC_HD_25_10_HEAAC_EBUTTD" type="video/mp4" transport="dash"
               sync_tl="dash_pr"/>
<video_profile name="MP4_HEVC_HD_25_10_HEAAC_EBUTTD" type="video/mp4" />
```

and, if E-AC3 audio is supported in the broadcast channel, shall additionally include the following video profiles:

```
<video_profile name="MP4_HEVC_HD_25_10_E-AC3_EBUTTD" type="video/mp4"
               transport="dash" sync_tl="dash_pr"/>
<video_profile name="MP4_HEVC_HD_25_10_E-AC3_EBUTTD" type="video/mp4" />
```

The strings defined in Table 13 shall be used to indicate which options are supported by a terminal. They shall be used:

- In the HTTP `User-Agent` header for applications data retrieval through HTTP.
- In the `ui_profile` element's `name` property of the `xmlCapabilities` property of the `application/oipfCapabilities` embedded object.
- As parameters of the `hasCapability()` method of the `application/oipfCapabilities` embedded object to dynamically query the options supported by the terminal.

NOTE: Some of the strings defined in the clause intentionally match with the "UI Profile Name Fragment" strings defined in the OIPF DAE specification [1].

Table 13: HbbTV® Option Strings

Option string	Meaning
"+DL"	Support for file download feature.
"+PVR"	Support for PVR feature.
"+DRM"	Support for the DRM feature - specifically that the XML capabilities include a `<drm>` element as defined below (see note).
"+SYNC_SLAVE"	Support for the terminal behaving as a slave terminal for inter-device synchronization (see clause 10.2.9).
"+IPH"	Support for the "IP delivery Host player mode" as defined in the DVB Extensions to CI Plus TS 103 205 [37].
"+IPC"	Support for the "IP delivery CICAM player mode" as defined in the DVB Extensions to CI Plus TS 103 205 [37].
"+AFS"	Support for the CICAM Auxiliary File System as defined in the DVB Extensions to CI Plus TS 103 205 [37].
NOTE: "+DRM" has a specific meaning in OIPF which it does not have in the present document.	

The support of the DRM feature shall be indicated by the addition of one or more `<drm>` elements in the OIPF extension to the `<profilelist>` element as defined in annex F of the OIPF DAE specification [1] to the end of the `<profilelist>` element in the above XML. For example:

```
<drm DRMSystemID="urn:dvb:casystemid:12345">TS_PF</drm>
```

The support of one or more CA systems on a CICAM shall be indicated using the `<drm>` element defined in annex F of the OIPF DAE specification [1] and providing the `protectionGateways` attribute with "`ci+`" string. All of the CA systems exposed by the CICAM using the `ca_info` APDU shall be listed in this way. For example:

```
<drm DRMSystemID="urn:dvb:casystemid:12345" protectionGateways="ci+">TS_PF</drm>
```

Terminals that support the "IP delivery Host player mode", as defined in the DVB Extensions to CI Plus TS 103 205 [37], shall expose the DRMs supported by any compliant CICAMs inserted in the terminal by using the `<drm>` element as described above. In this case, the protection format shall be "`MP4_CENC`" although others may also be listed where appropriate. All of the DRM systems exposed by the CICAM using the `sd_info_`reply APDU shall be listed in this way. This implies that the CICAM shall identify a supported DRM by filling in the `drm_system_id` field in the `sd_info_reply` APDU. The URN string describing the DRMs shall be suffixed by "`:cicam`". For example:

```
<drm DRMSystemID="urn:dvb:casystemid:12345:cicam">MP4_CENC</drm>
```

10.2.5 Void

10.2.6 Parental access control

10.2.6.1 Broadcast channel

Terminals shall support parental access control for the broadcast TV content and application as required for the markets in which the products are to be sold or deployed. The details of this are outside the scope of the present document. Typically the end user may have to enter the appropriate PIN in order to obtain access to TV content above the parental rating threshold. The following shall apply if access to broadcast TV content and/or application is blocked as a result:

- If access to broadcast TV content is blocked when changing to a channel, this shall be reported to any running HbbTV® application which survives the channel change and has registered a listener for a `ChannelChangeError` event as an error with error state 3 ("parental lock on channel").
- If access to broadcast TV content becomes blocked while a channel is selected, this shall be reported to any running HbbTV® application which has registered a listener for a `ParentalRatingChange` event.

The following shall apply if access to broadcast applications is blocked as a result:

- If access to an application in the broadcast AIT is blocked and the launch of this application is due to the behaviour defined in clauses 6.2.2.2 and 6.2.2.3, then the rules defined in those clauses apply.
- If access to an application in the broadcast AIT is blocked and the launch of this application is being requested by another application, then this launch request shall fail and the application shall not be loaded or run.

In terminals where CI or CI Plus [12] is supported, the CICAM may also enforce parental access control for the broadcast channel.

10.2.6.2 Broadband delivered content

Applications offering access to streaming on-demand content shall obtain the parental rating system threshold set on the terminal and only stream appropriate content to the terminal.

In spite of the above, the present document requires the terminal to react to signaled parental rating information in some circumstances. Some combinations of system formats and media APIs permit parental rating information to be associated with broadband streamed content. Where the present document requires support for the location where that parental rating information may be carried and where parental rating information is provided in a parental rating scheme that the terminal supports then this shall be checked. Terminals shall not just ignore such information assuming that applications comply with the above requirement.

The content access descriptor has an optional `<ParentalRating>` element which can be used to carry parental rating information associated with the content that it references.

Particularly for live content, clause 9.1.2.3 of the DVB DASH profile TS 103 285 [45] defines how parental rating information can be carried using the DASH event mechanism. In markets where the DVB-SI parental rating descriptor is used in the broadcast, terminals shall react to these DASH events in the same way as they react to the DVB-SI parental rating descriptor in the broadcast channel. In markets where the DVB-SI parental rating descriptor is not used in the broadcast, terminals may enforce the parental rating information carried in these DASH events.

NOTE: It is implementation dependent whether presentation of content is paused while any parental access authorization is obtained (in which case presentation would start or resume behind the live edge) or if content is discarded (so that presentation starts or resumes at the live edge).

In terminals where CI or CI Plus [12] is supported, the CICAM may also enforce parental access control for streamed content.

10.2.6.3 Downloaded content

Broadcasters and service providers offering content for download shall populate the otherwise optional `<parentalRating>` element in the content access descriptor with the correct value for each content item downloaded. When playing back a downloaded content item, terminals shall compare the value in the `<parentalRating>` element in the content access descriptor used to download the content item with the current parental rating system threshold and only play appropriate content. In markets where the DVB-SI parental rating descriptor is used in the broadcast, terminals shall support the use of that parental rating scheme for downloaded content. In markets where the DVB-SI parental rating descriptor is not used in the broadcast, terminals should support an appropriate parental rating scheme for downloaded content.

NOTE: The definition of what content is appropriate is outside the scope of the present document. Typically this could be any content under the threshold or content above the threshold where the end-user has entered a PIN.

If playback which was initiated by an HbbTV® application is blocked following such a comparison:

- If the playback was initiated using an A/V Control object then the object shall enter play state 6 (error) with the `error` property set to 7 ("content blocked due to parental control") and an `onParentalRatingChange` event posted.
- If the playback was initiated using an HTML5 video element then the `error` property of the video element shall be set to a `MediaError` with the code set to `MEDIA_ERR_DECODE` and fire an error event at the media element.

In terminals where CI or CI Plus [12] is supported, the CICAM may also enforce parental access control for downloaded content.

10.2.6.4 PVR

Broadcasters and service providers whose applications create `Programme` objects and pass them to the `record(Programme programme)` method of the `application/oipfRecordingScheduler` object shall populate the `parentalRating` property of the `Programme` object. Terminals shall obtain the parental rating information from DVB-SI (or an alternative source if one is supported and in use) at the time of recording and store this with the scheduled recording in the system and copy it to the in-progress recording once the recording process starts. This shall override any parental rating information provided earlier in the recording process. Where a recording is scheduled using the `recordAt()` method or started using the `recordNow()` method, the parental rating assigned to the recording shall be the most restrictive value encountered during the recording process.

Before playing back a recording, terminals shall compare the parental rating stored with the recording with the current parental rating system threshold and shall only play appropriate content.

NOTE: The definition of what content is appropriate is outside the scope of the present document. Typically this could be any content under the threshold or content above the threshold where the end-user has entered a PIN.

If playback which was initiated by an HbbTV® application is blocked following such a comparison:

- If the playback was initiated using an A/V Control object then the object shall enter play state 6 (error) with the `error` property set to 7 ("content blocked due to parental control") and an `onParentalRatingChange` event posted.
- If the playback was initiated using an HTML5 video element then the `error` property of the video element shall be set to a `MediaError` with the code set to `MEDIA_ERR_DECODE` and fire an error event at the media element.

When playing back an in-progress recording, if the parental rating value of the recording changes, the terminal shall:

- Dispatch a `ParentalRatingChange` event.
- Compare the new parental rating value with the current parental rating threshold and, if the content has become inappropriate, the A/V Control object shall enter play state 6 (error) with the `error` property set to 7 ("content blocked due to parental control").

In terminals where CI or CI Plus [12] is supported, the CICAM may also enforce parental access control for recorded content.

10.2.6.5 Synchronization and parental access control

If access to media content presented by a video/broadcast object, an HTML5 media element or an A/V Control object is blocked by the terminal due to parental access control then:

- If it represents master media (it was passed to a `MediaSynchroniser` via the `initMediaSynchroniser()` method) then this shall result in a permanent error of the `MediaSynchroniser` (see clause 13.3.8) with an error being triggered with error number 14.
- If it represents other media (it was added to a `MediaSynchroniser` via the `addMediaObject()` method) then this shall result in a transient error of the `MediaSynchroniser` (see clause 13.3.7) and the object shall be removed as if an application had called the `removeMediaObject()` method and an error event triggered with error number 2.

10.2.7 Component selection

10.2.7.1 General

Where more than one component of a media type (i.e. video, audio, subtitles) is available, selection of which one or ones should be presented may be done either by the HbbTV® terminal or by the HbbTV® application or by both.

The terminal shall always perform a default selection of components to present from all of the available components. This is described in clause 10.2.7.2 below. A running HbbTV® application may override the default selection as described in clause 10.2.7.3 below. There is no linkage between how selection of different media types is done; an application may override the default selection for any one media type, any two media types or all or none to modify the presentation of the media to the user.

The set of components that are available for this selection depends on how media is being presented. Four scenarios are defined:

- A single presentation is in progress using a single media object. In this case, the components that are available are those found in the input or source to that media object. For an HTML5 media element, these are the `VideoTracks`, `AudioTracks` and `TextTracks`. For an A/V Control object or a video/broadcast object, these are the `AVComponents`.
- A single presentation is in progress using more than one media object synchronized using multi-stream synchronization as defined in clause 10.2.8 of the present document. In this case, the components that are available are the union of those in the master media object and all the other media objects attached to the `MediaSynchroniser` that were not added with the `multiDecoderMode` argument set to true.
- Multiple presentations are in progress using more than one media object synchronized using multi-stream synchronization as defined in clause 10.2.8 of the present document. The application can choose this type of presentation by adding media objects to the `MediaSynchroniser` with the `multiDecoderMode` argument set to true. In this case, the terminal treats each such media objects as single presentations.
- Multiple presentations are in progress using more than one media object without synchronization.

NOTE: Support for multiple presentations is optional in the present document and depends on the support for multiple decoders.

10.2.7.2 Component selection by the terminal

It is the responsibility of the terminal to choose for presentation to the user the most appropriate default components from those available in the media object(s), based on the user's preferences (e.g. audio language). The terminal shall present to the user the default components of those component types which are selected; this selection shall also be based on user preferences (e.g. subtitles on/off). The terminal shall take into account all components whether available on broadcast or broadband.

If the components available within a presentation change and selection of one or more media type is being done by the terminal, then the terminal may choose a component, previously not selected, for presentation to the user, for example if that component fits better with the user's preferences.

In particular, when a new media object is added to a `MediaSynchroniser`, the terminal shall re-evaluate the default selection of presented components and component types including all of the components that make up that media object (as well as the existing media objects added to the `MediaSynchroniser`). Likewise, the removal of a media object from a `MediaSynchroniser` shall cause the terminal to re-evaluate which components to be presented by default.

Terminals shall support a method for the user to enable and disable subtitles and to select at least one preferred subtitle language. Terminals shall use this information when playing content to determine whether to present subtitles and to select between multiple subtitles when they are available.

If display of subtitles is disabled using this method, use of the component selection API to select a subtitle component will not result in those subtitles being displayed.

NOTE 1: Applications may use the property `subtitlesEnabled` as defined in clause A.2.20 to check whether selecting subtitle components is currently possible or the user has to enable this through a terminal UI.

Terminals shall support a method for the user to enable and disable audio description streams as defined in clause 7.1.2 of the present document. Terminals shall use this information when playing content to determine whether to present audio description streams instead of the normal audio (or in addition to the normal audio where receiver mix audio description is supported).

This method may also be used to select other audio streams for example clean audio or alternative audio languages.

If either or both of the subtitle components or the audio description components available in the content change and a previously selected component is no longer available, then the terminal should re-evaluate the subtitle or audio description component selection as applicable based on the user preferences.

NOTE 2: Use of the terminal's audio description selection mechanism by the user may change the selected audio track. Applications using an HTML5 media element should register a listener for 'change' events on the AudioTrackList object if they need to be aware of such changes.

The terminal shall present to the user the default selection of components unless the application overrides it (see clause 10.2.7.3).

10.2.7.3 Component selection by the application

Applications may change the terminal-derived component selection and discover the presentation status using the methods defined in clause 7.16.5 of OIPF DAE [1] and in clauses 4.7.10.10.1 and 4.7.10.12.5 of HTML5 [54].

If selection of one or more media type has been done by the application on a media object and that media object is subsequently added to a `MediaSynchroniser` with the `multiDecoderMode` argument set to true, then the selected components shall continue to be selected as described in this clause.

If an application selects a new component for a media element that was added to a `MediaSynchroniser` using the single decoder model (`multiDecoderMode=false`), then the terminal shall unselect any component of the same type (audio, video or subtitles) previously presented by any media object that is part of the same single presentation of the `MediaSynchroniser` as defined in clause 10.2.7.1.

The terminal shall maintain such changes made by an application until one of the following occurs:

- the application terminates:
  - in which case component selection shall revert to the control of the terminal;
- the application makes a further change:
  - in which case the behaviour shall be as defined by the API where that change was made;
- a component, selected by the application, is being presented and is part of a video/broadcast object or an A/V Control object or an HTML5 media element or a `MediaSynchroniser` object (as appropriate) which is destroyed:
  - in which case component selection for that component type shall revert to the control of the terminal;
- 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;
- in the case of a video/broadcast object, a component, selected by the application from that video/broadcast object, is being presented and the broadcast channel is changed either by an application as defined in the present document or by a mechanism outside the scope of the present document (e.g. the end-user pressing P+ or P- on a remote control):
  - in which case component selection for that component type shall revert to the control of the terminal;
- the media object is added to a `MediaSynchroniser` with the `multiDecoderMode` argument set to false:
  - in which case all component selections on the media object shall be unselected and the component selection rules for the `MediaSynchroniser` shall apply.

NOTE: For on-demand content, the application may override the default selection before any content has been presented.

If both (a) a presentation involving multi-stream synchronization is either in progress or starting or stopping and (b) selection of one or more media type has been done by the application, then the selected component shall continue to be selected as long as it remains in the set of components available for selection.

EXAMPLE: If an application has selected a component in a media object which is made the master media object for multi-stream synchronization and one more other media objects are added to the synchronization then the original selection shall be retained as long as the selected component remains available. If the selected component ceases to be available then the behaviour shall be as defined by the API where that selection had originally been made.

10.2.7.4 Single decoder model

The rules defined in clauses 10.2.7.2 and 10.2.7.3 shall apply when the terminal has sufficient decoder resources to be able to independently present only one media object, where that media object could be an HTML5 media element, an A/V Control object, a video/broadcast object or a `MediaSynchroniser` object.

EXAMPLE: If an application uses the `MediaSynchroniser` to present a broadcast service with one video, one audio and one subtitle component and a broadband stream with one audio and one subtitle component, the terminal may choose for the presentation the video and subtitle components from broadcast and the audio component from broadband. This may be based on user preferences set in the terminal. The application may then select also the subtitle component from broadband. As the presentation for both media objects follows the single decoder model, the subtitle component from broadcast will be unselected before the broadband subtitles are presented to the user.

10.2.7.5 Multi-decoder model

When the terminal has sufficient decoder resources to be able to independently present more than one media object (where that media object could be an HTML5 media element, an A/V Control object or a video/broadcast object), the rules defined in clauses 10.2.7.2 and 10.2.7.3 shall apply to each media object separately.

In the case of a `MediaSynchroniser` object, any media object that is added to the `MediaSynchroniser` with the `multiDecoderMode` argument set to true shall be treated as being an independent media object as far as this clause is concerned.

In the case of a `MediaSynchroniser` object, any media object that is added to the `MediaSynchroniser` with the `multiDecoderMode` argument set to false shall be treated in combination with all other media objects added in the same way and with the master media object as being an independent media object as far as this clause is concerned.

EXAMPLE: If an application uses the `MediaSynchroniser` to present a broadcast service with one video, one audio and one subtitle component and a broadband stream with one video, one audio, one subtitle component that was added with the `multiDecoderMode` set to true, the terminal will choose the components for each media object independently. If subtitles are enabled, the terminal will choose the video, audio and subtitles components from both media objects and present all of them. The application may then unselect the subtitle component from broadband. As the presentation for both media objects follows the multi decoder model, the subtitle component from broadcast will not be unselected.

10.2.8 Multi-stream media synchronization

10.2.8.1 General

The HbbTV® terminal shall support decoding and rendering of A/V content delivered as multiple streams over broadband and broadcast networks as defined in this clause. This capability is known as multi-stream synchronization and clause 13.2 describes an architecture for it.

The HbbTV® terminal shall support the combination of at least one broadcast service and at least one broadband stream. A broadcast service can be any DVB service supported by the HbbTV® terminal. Formats for supported broadband streams are defined below in clause 10.2.8.2 and clause 10.2.8.3, with constraints defined in clause 10.2.8.4. An EBU-TT-D file downloaded out of band does not count as a broadband stream for the purposes of this clause.

NOTE 1: One use-case for multi-stream sync is broadband delivery of additional audio languages or accessible audio streams to be synchronized with broadcast video. When making a scheduled recording, PVRs will not have the information to record this broadband delivered audio along with the broadcast video and audio. Broadcasters should consider this when deciding whether to send additional audio in the broadcast as normal or whether to send it via broadband and use multi-stream sync to combine it with the video.

Applications can use the `maxBroadbandStreamsWithBroadcast` and `maxBroadbandStreamsNoBroadcast` properties of the `MediaSynchroniser` object to determine the maximum number of broadband streams that can be used in combination for multi-stream synchronization.

EXAMPLE: If the terminal supports only the combinations of streams listed in Table 14, then both `maxBroadbandStreamsNoBroadcast` and `maxBroadbandStreamsWithBroadcast` properties will have the value 1. If, however, the terminal supports combinations where 2 broadband streams can be synchronized (such as video in an MPEG2-TS via broadband and audio in an MPEG DASH presentation) then the `maxBroadbandStreamsNoBroadcast` property instead has the value 2. Also, if the terminal supports combinations where broadcast can be synchronized with 2 broadband streams (such as video via broadcast and audio in a DASH presentation and an additional video via MPEG2-TS via broadband) then the `maxBroadbandStreamsWithBroadcast` property instead has the value 2.

Broadcast services and broadband streams usually consist of multiple video, audio and subtitle components. The HbbTV® terminal shall support simultaneous decoding and rendering of at least one video component, at least one audio component, and one subtitle component. Each component may come from any of the input streams. The HbbTV® application may use an API to select components from the input streams as defined by the APIs profiled in annex A.

The presentation of two or more components of the same type is optional. Applications can use the `extraHDVideoDecodes` property as defined in clause 7.15.3.1 of the OIPF DAE specification [1] to check for additionally available video and audio decoders.

The HbbTV® terminal shall support the synchronization methods as defined in clauses 10.2.8.2 and 10.2.8.3 to synchronize the presentation of components from multiple streams.

The terminal shall implement the buffer model as defined in clause 13.5 to handle the different transmission delays of each stream.

The terminal shall implement the `MediaSynchroniser` API defined in clause 8.2.3.

NOTE 2: The broadcaster has to ensure that the delivery of the streams is in time for synchronized presentation, in order to prevent synchronization buffer overflows. Clause G.2 gives guidance to broadcasters how to minimize the delay.

When an application adds a new stream for synchronization to already presenting media object(s) (broadcast services or broadband streams) the terminal shall adjust the presentation timing of some or all of the streams to attempt to synchronize them according to a timing relationship expressed by the application. This timing relationship is expressed by identifying a point X on the timeline of the existing master media stream (see clause 13.2.4) and a point Y on the timeline of the new stream that are to be co-timed within a specified tolerance.

Any difference in timing of presentation for streams that are synchronized according to the timing relationship shall be no greater than plus or minus the greater of: the application specified tolerance and the minimum synchronization accuracy defined in clause 9.7.4.

When the terminal attempts to achieve synchronization between the streams using a specified timing relationship, the new stream may be behind (in the past) or ahead (in the future) by N seconds compared to the master media stream. The terminal can employ various strategies to achieve synchronization, including:

- pausing the presentation of the new or existing streams temporarily for up to N seconds until they can be resumed in synchronization;
- or jumping backward or forward in the new or existing streams by up to N seconds and continuing their presentation (using its own buffer or a network-based buffer, like a CDN, a RET server or an FCC server).

The terminal can employ strategies in combination, for example: jumping forward in the existing streams and pausing the new stream temporarily to delay it.

The terminal shall also adjust the timing relationship of some or all of the streams if the application specifies a new timing relationship for a stream. The terminal can employ strategies in the same way it would for a new stream, as described above. Any differences in timing of presentation between the streams shall remain no greater than plus or minus the greater of: the application specified tolerance and the minimum synchronization accuracy defined in clause 9.7.4.

For a more detailed explanation of the above process, refer to clause C.3 of TS 103 286-2 [47].

When an application adds a stream for synchronization to an already presented broadcast service or broadband stream the terminal may either pause the presented media object or rewind in that stream to get a faster start of the synchronized presentation.

The present document does not require HbbTV terminals to support multi-stream media synchronisation for content played by CICAM player mode as defined in clause 11.4.5 and annex K of the present document.

10.2.8.2 Synchronization using gen-locked STC

This clause applies to the synchronization of one broadcast service with at least one broadband stream, where the broadband streams are MPEG transport stream based and share the STC with the broadcast service.

NOTE: Two streams sharing the same STC means that the encoder clocks of both are gen-locked and that the PTS of all streams are derived from the same STC. The terminal can assume that there is no drift between the clocks of the broadcast service and broadband stream. The PCR/PTS of the streams are not modified in the delivery chain, e.g. by the network operators in case of transcoding.

If the terminal supports the buffer for media synchronization as defined in clause 13.5.3, the terminal shall support multi-stream media synchronization of a broadcast service with broadband streams delivered as SPTS as defined in clauses 7.3.1.2 and 7.3.2.1. In this case:

- the terminal shall use the PCR/PTS of the broadcast service as the timeline as defined in clause 13.4.2, referred to as MPEG-2 TS Presentation Timestamps (PTS);

- the terminal shall support broadband streams that do not contain a PCR. The terminal shall use the PCR of the broadcast service also as the timeline for the broadband stream.

10.2.8.3 Other synchronization cases

This clause applies to the synchronization of one broadcast service with one or more broadband streams where the streams may have different system formats or different types of timelines. In case the broadband stream is MPEG transport stream based, its STC can be independent from the STC of the broadcast service.

Terminals should support multi-stream synchronization for all combinations of system formats and types of timelines as defined in this clause. Terminals shall at least support the combinations listed as mandatory in clause 10.2.8.4.

Terminals shall support multi-stream synchronization for broadcast services with timelines defined for MPEG-TS in clause 13.4.2 and the constraints defined in clause 10.2.8.4.

Terminals shall support multi-stream synchronization with the constraints defined in clause 10.2.8.4 for broadband streams which are:

1) delivered as SPTS as defined in clauses 7.3.1.2 and 7.3.2.1 with the timelines defined for MPEG-TS in clause 13.4.2; or

2) delivered with MPEG DASH as defined in annex E with the timeline defined for MPEG-DASH in clause 13.4.2; or

3) delivered via HTTP as an EBU-TT-D compliant TTML document as defined in clause 7.3.1.5.1 with timeline as defined in clause 13.4.2.

Terminals may support multi-stream synchronization for broadband streams which are:

1) encapsulated in the ISOBMFF format as defined in clause 7.3.1.1 (referred to as "MP4") with the timeline defined for ISOBMFF in clause 13.4.2.

The relationship between the timelines will be provided by the application using the APIs defined in clause 8.2.3. The terminal shall implement the `MediaSynchroniser` API defined in clause 8.2.3.

10.2.8.4 Supported combinations

HbbTV® terminals shall support the presentation of at least the combinations of media type, systems layer, timeline and delivery protocol for multi-stream synchronization, as shown in Table 14.

NOTE: Table 14 does not apply when inter-device synchronisation is being performed without multi-stream synchronisation.

All combinations apply only to the codecs which are mandatory in the present document. If a combination has two audio or two video streams the terminal is not required to support streams with different video or audio codecs.

Support for multi-stream synchronization with E-AC-3 is required where E-AC-3 is supported via the broadcast connection, i.e. restrictions on codecs from clause 7.3.1.1 apply.

Table 14: Mandatory combinations of media type, systems layer, timeline and delivery protocol

Delivery	Broadcast		Progressive Streaming			MPEG DASH	HTTP Out of Band	Status
Systems Layer	MPEG2-TS		ISOBMFF	MPEG2-TS		ISOBMFF		
Timeline for m/s sync	PTS	TEMI	CTS	PTS	TEMI	DASH-PR	EBU-TT-D	
1		Video				Audio		M
2		Video				Audio	Subtitles	M
2a		Video, Audio					Subtitles	M
3		Video, Audio				Subtitles		M
4		Video, Subtitles				Audio		M
5		Video				Audio, Subtitles (see note 2)		M
6		Video, Audio				Video		M-2V
7		Video				Video, Audio (see note 2)		M-2V
8	Video (see note 3)			Audio (see note 3)				M-SB
9	Video (see note 3)			Audio, Subtitles (see notes 1 and 3)				M-SB
10	Video, Audio (see note 3)			Subtitles (3)				M-SB
11	Video, Audio (see note 3)			Video (see note 3)				M-SB, M-2V
12	Video (see note 3)			Video, Audio (see note 3)				M-SB, M-2V

NOTE 1: Delivered in the same MPEG2-TS.
NOTE 2: Delivered in the same MPEG-DASH session.
NOTE 3: Both with gen-locked STC (see clause 10.2.8.2) and without gen-locked STC (see clause 10.2.8.3).

Table 15: Key to status column

Status	Meaning
M	Mandatory.
M-SB	Mandatory if the terminal implements the synchronization buffer as defined in clause 13.5.
M-2V	Mandatory if the terminal supports the simultaneous use of two video decoders for HbbTV® services.
NOTE: If the status column lists more than one conditional feature the combination is only mandatory if the terminal implements all the listed features.	

HbbTV® terminals are not required to support receiver mix audio description when the main audio and the audio description are delivered via different routes or in separate ISOBMFF files, MPEG2-TS streams or MPEG-DASH sessions.

NOTE: Support for synchronization involving encrypted broadband-delivered content is outside the scope of the present document and may be specific to the content protection technology involved.

10.2.9 Inter-device media synchronization

10.2.9.1 General

The HbbTV® terminal shall support decoding and rendering of A/V content delivered by broadcast or broadband networks in a manner where presentation is time synchronized with another HbbTV® terminal or a Companion Screen application. This feature is known as inter-device synchronization.

Clause 13.2 describes an architecture for inter-device synchronization and describes the roles of a master terminal and a slave terminal. A terminal shall be capable of being a master terminal and may optionally capable of being a slave terminal.

NOTE: Slave terminals and Companion Screen applications connect to interface endpoints provided by the master terminal and communicate with it via these interfaces using the protocols described in clause 13. The master terminal decides the timing of presentation of the master terminal, the slave terminals and Companion Screen applications. The slave terminals and Companion Screen applications adjust to follow the presentation timing recommended by the master terminal.

Clause 13.10 provides sequence diagrams illustrating the relationship between the `MediaSynchroniser` APIs and the inter-device synchronization protocols discussed in clauses 13.6, 13.7 and 13.8.

The present document does not require HbbTV terminals to support inter-device media synchronisation for content played by CICAM player mode as defined in clause 11.4.5 and annex K of the present document.

10.2.9.2 Master terminal

An HbbTV® terminal shall be able to act in the role of a master terminal. To implement this:

- The terminal shall implement the `MediaSynchroniser` API defined in clause 8.2.3.
- The terminal shall support generation of timestamps as defined in clause 13.4.1 and should support interpretation of timestamps as defined in clause 13.4.1.
- The terminal shall derive timelines defined in clause 13.4.2 from media streams as directed by an HbbTV® application via the `MediaSynchroniser` API.
- The terminal should implement the buffering model defined in clause 13.5 for inter-device synchronization.

NOTE: Some aspects of this buffer model are required for multi-stream synchronization. and these are defined in more detail in clause 13.5.

- The terminal shall implement the functions and interfaces for a master terminal described in clauses 13.6.2, 13.7.2, 13.7.3 and 13.8.2.

10.2.9.3 Slave terminal

An HbbTV® terminal may have the capability to act as a slave terminal, and this shall be signalled by the inclusion of the appropriate option string defined in Table 13 in clause 10.2.4.

For a terminal with the capability to act as a slave terminal:

- The terminal shall implement the functionality and interfaces for a slave terminal described in clauses 13.6.3, 13.7.4 and 13.8.3.
- The terminal shall interpret Correlation Timestamps according to clause 13.4.3.
- The terminal shall support discovery of other HbbTV® terminals using the `discoverTerminals()` method.

NOTE: Some aspects of the Correlation Timestamp interpretation are required for multi-stream synchronization.

These requirements are in addition to the implementation requirements defined in clause 10.2.9.2.

10.2.10 Application to media synchronization

The terminal shall implement application to media synchronization as described in clause 13.11. The terminal shall support all combinations of types of timeline and types of content defined in 13.4.2.

10.2.11 Combining audio from memory and broadcast audio / video

Terminals shall support playing audio from memory via the Web Audio API simultaneously with playing Audio delivered via broadcast (either with or without video). Terminals should support playing audio from memory via the Web Audio API simultaneously with audio (with or without video) in all of the following cases;

- The main audio delivered via broadband
- The main audio was downloaded using the download feature (if supported by the terminal)
- The main audio was recorded using the PVR feature (if supported).

Terminals should seamlessly mix the audio from memory with the audio from the A/V and output the results on all outputs from the terminal. If this is not done then either or both of the following approximations shall be followed.

- Replacing the audio from the A/V with the audio from memory.
- Only outputting the audio from memory (either mixed or replacing the audio from A/V) on stereo outputs (e.g. built-in audio outputs of the HbbTV terminal) and not on multi-channel outputs (e.g. outputs to external audio systems such as home cinema systems or sound bars such as S/PDIF).

NOTE: These permitted approximations will be removed in a future version of the present document.

The following shall apply regardless of whether the two audio sources are mixed properly or if either or both of the approximations are followed.

- The playback of the audio from memory shall not disturb the playback of the video from the main A/V, e.g. the video shall not be paused, no frames shall be dropped and no black frames shall be inserted.
- Where an HbbTV terminal supports bitstream output(s) (e.g. S/PDIF), the terminal shall not change the number of channels in a bitstream output at the start or end of outputting audio from memory.

NOTE: Changing the number of channels normally gives a poor user experience and short clips of audio from memory may be completely lost in the time taken for the change to happen.

11 Security

11.1 Application and service security

The present document defines two levels of trust for applications - trusted and not trusted. The features only available to trusted applications are listed in Table A.1.

By default, broadcast related applications shall be trusted and broadcast-independent applications shall not be trusted. This may be modified as follows:

- Terminals may include a mechanism to allow the end-user to configure specific broadcast-independent applications as trusted or to configure broadcast-related applications from a particular service or channel as not being trusted.
- Terminals supporting reception of non-regulated channels should not automatically trust all applications from those channels.

EXAMPLE 1: In terminals supporting reception of satellite channels, for example, HbbTV® applications from adult channels on satellite should not be trusted except following explicit end-user approval and in compliance with appropriate regulation.

EXAMPLE 2: In terminals supporting reception of cable or terrestrial channels, if the markets addressed have the possibility of local or community access channels then HbbTV® applications from these channels are not required to be trusted.

The details of how regulated and non-regulated channels are identified are outside the scope of the present document.

- Terminals supporting cable or terrestrial reception of HbbTV® applications are not required to automatically trust all applications from all channels if different regulatory requirements apply to different channels. For example, HbbTV® applications from lightly or non-regulated local or community access channels which may be found in some markets are not required to be trusted. The details of how this could be achieved are outside the scope of the present document.
- Manufacturers may be able to configure specific broadcast-independent applications as being trusted and specific broadcast-related applications as being not trusted.
- Local regulation may impose additional requirements.

The security and permission mechanisms defined in clause 10.1 of the OIPF DAE specification [1] are not included in the present document. If they are included in a particular implementation then permissions should only be granted to an application where all mandatory parts of the feature or API covered by the permission are available.

NOTE: The set of features defined as available to trusted applications in the present document cannot be perfectly mapped onto the permissions defined in the OIPF DAE specification [1].

11.2 TLS and Root Certificates

11.2.1 TLS support

HTTP over TLS as defined in IETF RFC 2818 [7] shall be supported for transporting application files over broadband.

TLS version 1.2 as defined in IETF RFC 5246 [8] shall be supported. Terminals shall not set the `client_version` field of the `ClientHello` message to less than { 3, 3 } (TLS 1.2).

Terminals shall not negotiate sessions using SSL 3.0 or earlier.

Terminals shall support the Renegotiation Indication extension defined in IETF RFC 5746 [58].

Terminals shall support the Server Name Indication extension defined in IETF RFC 6066 [59].

Terminals shall support the Supported Elliptic Curves extension defined in IETF RFC 4492 [55]. See also clause 11.2.2.

In accordance with TLS 1.2, terminals shall indicate the supported signature algorithms using the Signature Algorithms extension. See also clause 11.2.4.

For optimal performance, terminals should securely maintain a TLS session cache and attempt to resume previously-established TLS sessions where possible. Terminals should support the Session Ticket extension defined in IETF RFC 5077 [56].

NOTE 1: Session resumption can significantly reduce the overhead of establishing a new TLS session with a recently-used server.

NOTE 2: Security considerations for caching and resuming sessions can be found in clause F.1.4 of IETF RFC 5246 [8].

Terminals shall not use TLS-level compression.

Terminals shall deem a TLS connection to have failed if any of the following conditions apply:

- The certificate chain fails validation as per section 6 of IETF RFC 5280 [9].
- Any signature required for certificate chain validation uses an algorithm or key size that is forbidden by the present document.

NOTE 3: This requirement relates only to signatures that are actually required to be verified and does not cover signatures on root certificates or signatures on any additional certificates presented by the server for compatibility with older clients.

- The host name or IP address contained in the server certificate does not match the host name or IP address requested. When verifying the host name against the server-supplied certificate, the '*' wildcard and the `subjectAltName` extension shall be supported as defined in IETF RFC 2818 [7].

Terminals shall not provide the user with an option to bypass these conditions.

11.2.2 Cipher suites

The cipher suite requirements are specified in Table 15a. The cipher suites are defined in IETF RFC 5246 [8] and IETF RFC 5289 [57]. Terminals should prioritize these cipher suites in the order shown. Terminals shall implement all cipher suites marked mandatory and shall not implement any cipher suites marked forbidden.

Table 15a: Cipher suites and their status

Cipher suite	Status
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	Mandatory
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	Mandatory
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	Recommended (see note)
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	Recommended (see note)
TLS_RSA_WITH_AES_128_CBC_SHA	Mandatory
Cipher suites with anonymous key exchange	Forbidden
Cipher suites with NULL encryption	Forbidden
Cipher suites using RC4 encryption	Forbidden
Cipher suites using encryption or signing algorithms offering less than 112 bits of security	Forbidden
NOTE: Cipher suites with a 128-bit security level are considered adequate at the time of writing. However, in the event of any significant advances in cryptanalysis of AES or SHA-256 within the lifetime of a terminal conforming to the present document, a terminal also supporting the AES_256 cipher suites may retain the ability to establish a secure connection when a terminal supporting only the AES_128 suites may not.	

Servers should use one of the above ECDHE cipher suites in preference as these provide forward secrecy and improved security against certain attacks.

11.2.3 Root certificates

A list of root certificates is maintained at http://www.hbbtv.org/spec/certificates.html. The policy by which this list has been derived is outlined in annex D.

Terminals shall trust all root certificates identified as mandatory and may support those certificates identified as optional on that list, subject to the conditions in this clause.

Terminals should not trust any other root certificates.

NOTE 1: Including root certificates that are not on the list increases the risk of a man in the middle attack if those root certificates have not been audited to a similar or greater level than those on the list.

Terminals shall not trust any root certificate with a public key where the number of bits of security provided by the algorithm is less than 112 bits, as defined by clause 5.6.1 of [49].

NOTE 2: For RSA keys, this implies a minimum key size of 2 048 bits.

Terminals shall support a means by which the device manufacturer can remove or distrust root certificates after manufacture. This may be handled either via a firmware upgrade mechanism or preferably via a specific root certificate update mechanism that could allow more timely updates.

A manufacturer may choose to remove or distrust a mandatory root certificate in the terminal in response to a security threat.

Terminals should support a means of securely adding new root certificates after manufacture in order to maintain interoperability with servers over time.

11.2.4 Signature algorithms

The algorithm requirements for signature verification are specified in Table 15b.

Terminals shall not trust any signature that uses an algorithm designated as forbidden.

Terminals shall cease to trust any signature that uses SHA-1 as the digest algorithm after 31st December 2016.

Table 15b: Signature algorithms and their status

Algorithm name	TLS 1.2 identifier	Status
md5WithRSAEncryption	0x0101	Forbidden
sha1WithRSAEncryption	0x0201	Mandatory until forbidden by SHA-1 sunset requirement specified above.
sha256WithRSAEncryption	0x0401	Mandatory
sha384WithRSAEncryption	0x0501	Mandatory
sha512WithRSAEncryption	0x0601	Optional
ecdsa-with-SHA1	0x0203	Optional until forbidden by SHA-1 sunset requirement specified above.
ecdsa-with-SHA256	0x0403	Mandatory
ecdsa-with-SHA384	0x0503	Mandatory
ecdsa-with-SHA512	0x0603	Optional

11.2.5 Key sizes and elliptic curves

Terminals shall support RSA keys with modulus size between 2 048 and 4 096 bits.

Terminals shall not trust RSA signatures that are less than 2 048 bits in size.

The requirements for elliptic curves are specified in Table 15c. The curves are defined in IETF RFC 4492 [55]. Curves marked mandatory shall be supported for signature verification and key exchange.

Table 15c: Elliptic curves and their status

Curve name	TLS 1.2 identifier	Status
secp256r1 (NIST P-256)	0x0017	Mandatory
secp384r1 (NIST P-384)	0x0018	Mandatory
secp521r1 (NIST P-521)	0x0019	Optional

11.2.6 Backward compatibility

Service providers should be aware that earlier versions of the present document contained different mandatory TLS versions, extensions and cipher suites.

Where TLS servers need to retain support for terminals conforming to earlier versions, they should:

- be able to negotiate a TLS session using TLS 1.0, 1.1 or 1.2;
- be aware that the Server Name Indication extension may not be present;
- be prepared to use the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite if none of the ECDHE cipher suites listed in clause 11.2.2 is offered by the terminal.

If necessary, service providers can also direct applications to use different endpoints when running on terminals conforming to earlier versions of the present document.

11.3 TLS client certificates

In HTTP over TLS, the use of a client certificate authenticates the client to a service provider. Some business models require that an HbbTV® application is delivered exclusively to trusted HbbTV® terminal implementations. To support these, terminals may support use of client certificates.

Negotiation and delivery of client certificates to the server is defined by the TLS specification [8].

Client certificates shall comply with IETF RFC 5280 [9].

The provision of client certificates is outside the scope of the present document.

11.4 CI Plus

11.4.1 CI Plus communication

Terminals supporting CI Plus for protected content via broadcast shall support the following mapping from the application/oipfDrmAgent embedded object to the CI Plus protocol as defined by clause 4.2.3 "CI+ based Gateway" of the OIPF CSP specification [5]:

- 4.2.3.1 Mandatory.
- 4.2.3.2 Mandatory.
- 4.2.3.3 Mandatory.
- 4.2.3.4 Mandatory, except for clauses 4.2.3.4.1.2 and 4.2.3.4.3 which are Not Included.
- 4.2.3.5 N/A.
- 4.2.3.6 Not Included.
- 4.2.3.7 Mandatory using URI (Usage Rule Information) as defined in clause 5.7 of CI Plus [12] if the PVR feature is supported otherwise Not Included.
- 4.2.3.8 Mandatory using URI (Usage Rule Information) as defined in clause 5.7 of CI Plus [12] if the PVR feature is supported otherwise Not Included.
- 4.2.3.9 Not Included.
- 4.2.3.10 N/A.

Terminals supporting CI Plus shall accept CI Plus CICAMs that do not support the OIPF extensions defined by clause 4.2.3 'CI+ based Gateway' of the OIPF CSP specification [5]. Specifically, the failure for any reason to set up the SAS connection with the Open IPTV Forum `private_host_application_ID` shall not stop other CI Plus functionality, that does not depend upon this connection, from working normally.

Terminals supporting an embedded CA solution should support a mapping from the `application/oipfDrmAgent` to the embedded CA system to provide the same functionality as defined above.

11.4.2 IP delivery Host player mode

11.4.2.1 Error handling in "IP delivery Host player mode"

Terminals supporting the "IP delivery Host player mode" as defined in the DVB Extensions to CI Plus TS 103 205 [37] shall map the following values of `drm_status` in the `sd_start_reply` APDU to the `errorState` argument of `onDRMRightsError`.

Table 16: onDRMRightsError errorState values

drm_status	errorState
Decryption possible (0x00)	valid license, consumption of the content is unblocked (2)
Error - no entitlement (0x02)	no license, consumption of the content is blocked (0)

If the `sd_start_reply` APDU contains a non-zero value for transmission_status (indicating that an error has occurred), then:

- if the content is being presented by an A/V Control object then the object shall transition to the 'error' state;
- if the content is being presented by an HTML5 media element then this shall be reported as defined in clause 9.6.7 of the present document.

11.4.2.2 DRM metadata source

Table 34 in clause 7.4.4 of the DVB Extensions to CI Plus TS 103 205 [37] identifies 7 sources for DRM metadata. Of those 7 sources, the present document requires support for 0x03 "Common Encryption (CENC)" and 0x04 "Media Presentation Description (MPD)". The present document is intentionally silent about support for the other sources listed in that table.

11.4.3 Auxiliary file system

When the CICAM Auxiliary File System is implemented as specified in the DVB Extensions to CI Plus TS 103 205 [37], the Terminal shall declare the Auxiliary File System resource identifier in the list of the resources that it provides.

The HbbTV® Application Domain is defined as: "`HbbTVEngineProfile1`", i.e. the value of the "`AppDomainIdentifier`" and "`DomainIdentifier`" is "`HbbTVEngineProfile1`".

When the HbbTV® terminal receives a `FileSystemOffer` APDU with the `DomainIdentifier` set to "`HbbTVEngineProfile1`", the HbbTV® terminal shall acknowledge the `FileSystemOffer` by sending a `FileSystemAck` APDU with the `AckCode` set to 0x01.

11.4.4 Virtual channel

When the terminal supports the CICAM Virtual Channel as specified in clauses 14 and 15.3 of the DVB Extensions to CI Plus TS 103 205 [37], the Terminal shall support launching an HbbTV application when the virtual channel is selected. The details of how the HbbTV application is launched shall be as defined in clauses 4.5.2 and 5.4 of the "Content Security Extensions to the Common Interface" [63].

NOTE: Clause 4.5.2 of "Content Security Extensions to the Common Interface" [63] requires that “The Host shall give priority to broadcast applications as defined in clause 12.4.4.2 of TS 103 205 [37].”

11.4.5 IP Delivery CICAM player mode

Terminals supporting the "IP delivery CICAM player mode" as defined in the DVB Extensions to CI Plus TS 103 205 [37] shall support the use of “Host-initiated playback” (as defined in clause 8.3.2 of [37]) in combination with the HTML5 video element as defined in annex K of the present document.

11.5 Protected content via broadband

Terminals that support the "IP delivery Host player mode" as defined in the DVB Extensions to CI Plus TS 103 205 [37] shall support the decryption of protected content delivered via the broadband channel as defined in clause 7 of the DVB Extensions to CI Plus TS 103 205 [37] where that content is provided in an ISO base media file format, encrypted using MPEG common encryption as defined by CENC ISO/IEC 23001-7 [30] and constrained by annex B of the present document, and delivered using MPEG DASH (as defined in clause 7.3.2.1).

Support for the other features specified in the DVB Extensions to CI Plus TS 103 205 [37] are not required by this clause, unless there is a dependency from the referenced clause 7 of TS 103 205 [37].

Where a terminal supports the "IP delivery Host player mode", it shall be able to offer Representations to a CICAM where the UUID urn in the `@schemeIdUri` in a `ContentProtection` descriptor in the `AdaptationSet` containing the Representation matches a UUID in the `sd_info_reply` APDU returned by the CICAM to the terminal. This implies that the CICAM shall identify a supported DRM by filling in the `drm_uuid` field in the `sd_info_reply` APDU.

NOTE: Whether a terminal actually offers a Representation to a CICAM depends on which Adaptation Sets are in the MPD, on the DASH player algorithm for selecting between Adaptation Sets and on any explicit choice of Adaptation Sets by an HbbTV® application.

For terminals that do not support the "IP delivery Host player mode" as defined in the DVB Extensions to CI Plus TS 103 205 [37], support for decrypting content delivered via the broadband channel is optional in the present document. When decryption is supported via the integration of HbbTV® with one or more embedded content protection technologies, the terminal shall support at least the ISO base media file format using MPEG common encryption as defined by CENC ISO/IEC 23001-7 [30] and constrained by annex B of the present document as a format for encrypted content.

11.6 Protected content via download

Terminals that support the "IP delivery Host player mode" as defined in the DVB Extensions to CI Plus TS 103 205 [37] and also the download optional feature shall support the decryption of protected content as defined in clause 7 of the DVB Extensions to CI Plus TS 103 205 [37] where that content is provided in an ISO base media file format, encrypted using CENC (as defined CENC ISO/IEC 23001-7 [30] and constrained by annex B of the present document). This requirement shall apply if the content has been downloaded as a file from broadcast (delivered using FDP as defined in annex H) or from broadband.

Support for the other features specified in the DVB Extensions to CI Plus TS 103 205 [37] are not required by this clause, unless there is a dependency from the referenced clause 7 of TS 103 205 [37].

For terminals that do not support the "IP delivery Host player mode" in CI Plus but do support the download feature, support for decrypting protected content acquired using a download API is optional in the present document. When decryption is supported via the integration of HbbTV® with one or more embedded content protection technologies the terminal shall support at least the ISO base media file format used with MPEG common encryption as defined by CENC ISO/IEC 23001-7 [30] and constrained by annex B of the present document as a format for encrypted content.

11.7 Terminal WebSocket service endpoints

All WebSockets endpoint URLs for application to application communication and inter-device synchronization shall include a randomly-generated part. The URLs shall be static for at least the lifetime of an HbbTV® application and shall be regenerated every time the terminal starts. The URLs should have at least 128 bits of entropy.

NOTE 1: These requirements aim to make the endpoints hard to discover without using the specified discovery mechanism or the relevant HbbTV® APIs.

NOTE 2: See clause 14.5.1 for requirements on when the WebSocket server is required to accept connections and when it may reject them.

11.8 Cookie storage

Content providers may use cookies as part of a content protection solution and may include tokens that they do not want to be user-accessible or transferrable between devices.

The following requirements apply to HTTP cookies that have the Secure attribute set as defined in RFC 6265 [24] and are stored by the browser. NOTE: these requirements do not apply to cookies stored by media player functions.

- Terminals shall not provide any user-accessible feature that allows the user to inspect the contents of such cookies. NOTE: Terminals may still reveal their existence and their associated domains, and allow them to be deleted by the user.
- Such cookies shall only be transmitted over a TLS connection.
- Terminals should store such cookies in encrypted form using a device-unique key of strength equivalent to at least 128-bit AES.

NOTE: Content providers may be forced to block access to certain content from terminals that are found to have insufficient protection for stored data.

12 Privacy

12.0 Overview

This clause addresses privacy related terminal functions as well as privacy related measures on the application level.

12.1 Terminal privacy features

12.1.1 Tracking preference expression (DNT)

12.1.1.0 Background

The tracking preference expression mechanism defined in the present clause is a compatible subset of the W3C Working Draft for Tracking Preference Expression (DNT) [i.9]. It is intended that the present clause will be updated to reference that W3C specification, once it has been published as a W3C Technical Recommendation.

12.1.1.1 Principles

Clause 12.1.1.2 defines the `DNT` (do not track) header field for HTTP requests as a mechanism for expressing the user's preference regarding their behaviour being tracked (or not). The goal of this protocol is to allow HbbTV® terminals to express the user's personal preference regarding tracking to each server and web application that they communicate with via HTTP, allowing each service to either adjust their behaviour to meet the user's expectations, or to reach a separate agreement with the user to satisfy all parties.

To achieve this, any signal sent shall exclusively reflect the user's preference, not the choice of the terminal manufacturer, or any other mechanism outside the user's control. In the absence of user choice, legal, or regulatory requirements, no tracking preference shall be expressed by the terminal (i.e. the `DNT` header shall not be included in the HTTP request).

An HbbTV® terminal shall offer users a minimum of two alternative choices for a global Do Not Track preference: unset or `DNT:1`. A terminal may offer a third alternative choice: `DNT:0`. If the user's choice is `DNT:1` or `DNT:0`, the tracking preference is enabled; otherwise, the tracking preference is not enabled. A terminal may offer users additional Do Not Track preferences for finer grained control, for example to specify different behaviour for specific servers or web applications.

12.1.1.2 Expressing a tracking preference

12.1.1.2.1 Expression format

When a user has enabled a tracking preference, that preference needs to be expressed to all mechanisms that might perform or initiate tracking directly or by third parties, including sites that the HbbTV® terminal communicates with via HTTP.

When enabled, a tracking preference shall be expressed according to Table 17.

Table 17: Expression of tracking preference

DNT	Description
1	This user prefers not to be tracked on the target site.
0	This user prefers to allow tracking on the target site.

12.1.1.2.2 DNT header field for HTTP requests

HbbTV® terminals shall insert the `DNT` field into all outgoing HTTP requests made on behalf of an HbbTV® application as the means for expressing a user's tracking preference via HTTP.

NOTE 1: This does not apply to HTTP requests made by the media player or the DRM agent.

It shall be encoded as follows:

```
DNT-field-name  = "DNT"
```

```
DNT-field-value = ( "0" / "1" ) *DNT-extension
DNT-extension   = %x21 / %x23-2B / %x2D-5B / %x5D-7E
                ; excludes CTL, SP, DQUOTE, comma, backslash
```

Terminals shall send the `DNT` header field if (and only if) a tracking preference is enabled. Terminals shall not send the `DNT` header field if a tracking preference is not enabled. At most one `DNT` header can be present in a valid HTTP request.

EXAMPLE:

```
GET /something/here HTTP/1.1
Host: example.com
DNT: 1
```

The `DNT-field-value` sent by an HbbTV® terminal shall begin with the numeric character "1" (%x31) if all of the following conditions are met:

a) a tracking preference is enabled;

b) the preference is for no tracking;

c) there is not an exception for the origin server targeted by this request.

The `DNT-field-value` sent by an HbbTV® terminal shall begin with the numeric character "0" (%x30) if all of the following conditions are met:

a) a tracking preference is enabled;

b) the preference is to allow tracking in general or by specific exception for the origin server targeted by this request.

The remainder of the `DNT-field-value` after the initial character is reserved for future extensions. Terminals that do not implement such extensions shall not send `DNT-extension` characters in the `DNT-field-value`. Servers that do not implement such extensions may ignore any `DNT-extension` characters.

NOTE 2: The extension syntax is restricted to visible ASCII characters that can be parsed as a single word in HTTP and safely embedded in a JSON string without further encoding.

12.1.2 Third party cookies

Third party cookies are generally considered problematic in a privacy context. According to clause 7.1 of IETF RFC 6265 [24] the implementation of third party cookies is optional.

Manufacturers of HbbTV® terminals may block all third party cookies. If they do not, then they shall provide the user the option of doing so.

12.1.3 Blocking tracking websites

Tracking scripts can be problematic in a privacy context, especially when used in autostart applications. To provide additional protection to users, terminals may offer the possibility of blocking requests to tracking websites.

Manufacturers of HbbTV® terminals should consider providing the option of disallowing requests to tracking websites. If such an option is provided, manufacturers shall allow the user to set this option in a similar way to the DNT setting in clause 12.1.

The definition and maintenance of a corresponding list of sites to be allowed or disallowed remains the responsibility of each terminal manufacturer. Terminal manufacturers should take care that any such mechanism does not introduce privacy issues in its own right.

12.1.4 Persistent storage

Terminals may offer the user the option to disable persistent storage (cookies, Web Storage) on a per-application or per-site basis. While this may improve user privacy, it will likely result in a worse user-experience, for example loss of personalization and inability to remember a user's agreement to a site's terms and conditions. Persistent storage shall not be disabled by default.

12.1.5 Unique device IDs

Terminals may offer the user the option to disable the availability of a unique device ID (via the `deviceId` property defined in clause A.2.20.5 of the present document) on a per-application or per organisation basis. Access to the device ID should be enabled by default unless blocked due to local regulatory requirements.

It shall not be possible to determine the identifier that would be presented to one origin or device, knowing the identifier that was generated for a different origin or on a different device. The device ID shall be generated by deriving an ID of at least 128 bits using a secure hash function from a combination of a device unique value that is not required to be secret (e.g. serial number), plus a common secret value (e.g. common to a manufacturer or model or product family), plus the origin of the HTML document (see clause 6.3.2), plus a value that changes each time the user requests that a new value of the identifier is generated (e.g. the time the user request was made).

It shall be possible for the user to generate a new but distinct value for the device ID.

NOTE: This mechanism is modelled on the IOS 7 mechanism [i.14] and on the Android advertising identifier [i.15].

12.2 Respecting privacy in applications

Application developers are responsible to comply with all applicable legislation and regulation, particularly concerning user privacy.

Actions taken by broadcast-related autostart applications before the user has pressed the red button (or equivalent) are particularly sensitive since they occur unnoticed by the user, and hence without any possibility for him to intervene. Tracking or logging user data without prior given consent from the user is likely to breach almost any national privacy rules. Even where it would happen to be legal, such unnoticed and unexpected action is likely to be very controversial, and often prompts calls to consumers to refrain from connecting HbbTV® terminals to the Internet.

The present document provides a number of effective tools to meet such privacy requirements:

- delivering application data exclusively via DSM-CC (see clause 7.2.5) allows launching of applications without any data exchange via broadband, i.e. without sending any data to any server;
- use of TLS (see clause 11.2) to encrypt data exchanged via broadband;
- use of cookies (see clause 10.2.1) to record a user's consent to a service tracking or otherwise using or storing personal user data.

While these tools provide for a good, basic protection of user privacy, they cannot be guaranteed to meet all possible legal and regulatory obligations, and the present document shall hence not be construed as to make any assertions to this extent. Application developers are thus encouraged to perform a detailed analysis of any such legal and regulatory privacy obligations for each application, before releasing it into the market.

Application developers should further be aware that some features may be subject to user preferences as defined in clause 12.1.

13 Media synchronization

13.1 General (informative)

Clause 13 describes and defines how a terminal supports multi-stream and inter-device synchronization features and also the application to media synchronization feature. Clauses 10.2.8, 10.2.9 and 10.2.10 define the terminal requirements to implement these features.

Clause 13.2 describes a unified architectural model for both multi-stream synchronization and inter-device synchronization.

Clause 8.2.3 defines a single common API for HbbTV® applications to control the use of both these features.

Clause 13.3 describes the different states of media synchronization and their relationship to the API behaviour.

Clause 13.4 describes how timelines are derived from media streams for both these features and defines the reference point for measuring progress on the timeline for inter-device synchronization.

Clause 13.5 describes a buffering model for both these features.

Clauses 13.6 to 13.9 describe the functions and interfaces that enable inter-device synchronization.

Clause 13.10 shows sequence diagrams for APIs when used for inter-device synchronization.

Clause 13.11 describes application to media synchronization.

13.2 Architecture (informative)

13.2.1 General

The terminal performs multi-stream and/or inter-device synchronization functionality on behalf of the HbbTV® application. Both multi-stream and inter-device synchronization functionality are controlled by the HbbTV® application through the `MediaSynchroniser` object defined in clause 8.2.3.

The HbbTV® application directs the `MediaSynchroniser` object as to the streams to be rendered and the timing relationship between them.

NOTE: An HbbTV® application is typically expected to obtain information on the timing relationship from a Correlation Information Service (CIS), such as a Material Resolution Server (MRS) as described in TS 103 286-2 [47]. However whether or not a slave terminal communicates with an MRS is an implementation detail for the HbbTV® application and is outside the scope of the present document.

13.2.2 Multi-stream synchronization

Figure 17 illustrates the relationship between HbbTV® application and `MediaSynchroniser` object for multi-stream synchronization.

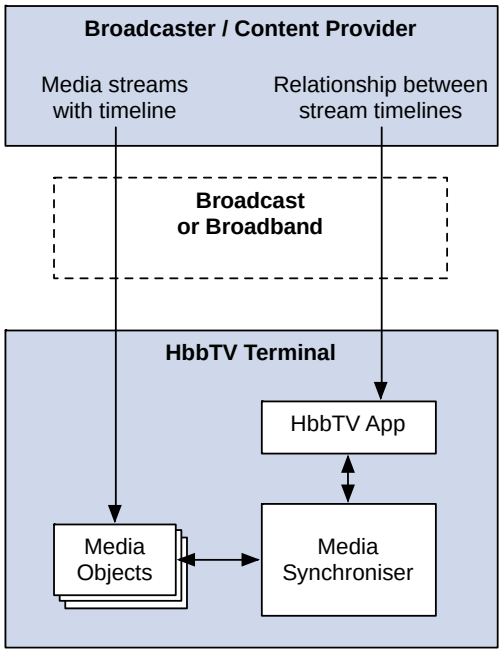


Figure 17: Relationship between MediaSynchroniser object and HbbTV® application for multi-stream synchronization

A terminal is deemed to be performing multi-stream synchronization when a `MediaSynchroniser` object is initialized and media objects are added to it using the `addMediaObject()` method. The terminal manages the decoding and rendering of the media streams.

Clause 4 of TS 103 286-2 [47] describes an architecture for synchronization that applies to both inter-device synchronization and multi-stream synchronization and defines the concepts of Media Synchronization Application Server (MSAS), Synchronization Client (SC), Correlation Information Server (CIS) and Material Resolution Server (MRS). Figure 18 illustrates the how these concepts apply to multi-stream synchronization.

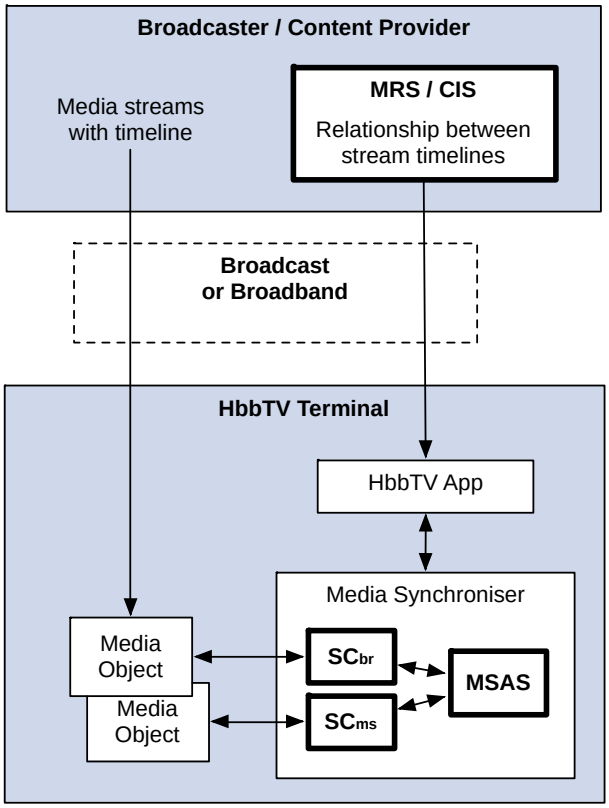


Figure 18: Basic mapping of media synchronization architecture for multi-stream synchronization

For multi-stream synchronization, the HbbTV® terminal and HbbTV® application running on it is equivalent to a single device containing multiple Synchronization Client (SC) elementary functions and the MSAS elementary function. Each SC elementary function manages the presentation of a single media stream such as broadcast (SCbr) or broadband media stream (SCms). The MSAS function is implemented by the `MediaSynchroniser` in the terminal.

13.2.3 Inter-device synchronization

Figure 19 and figure Figure 20 illustrate the relationship between `MediaSynchroniser` object and HbbTV® application for inter-device synchronization. A terminal is acting in the role of a master that dictates the timing of the presentation for all media streams. In figure Figure 19 a Companion Screen application is a slave whose timing is being dictated by the master. In figureFigure 20 the slave is another terminal.

Inter-device synchronization can happen simultaneously between a master terminal and one or more slave terminals and/or one or more Companion Screen applications on Companion Screens.

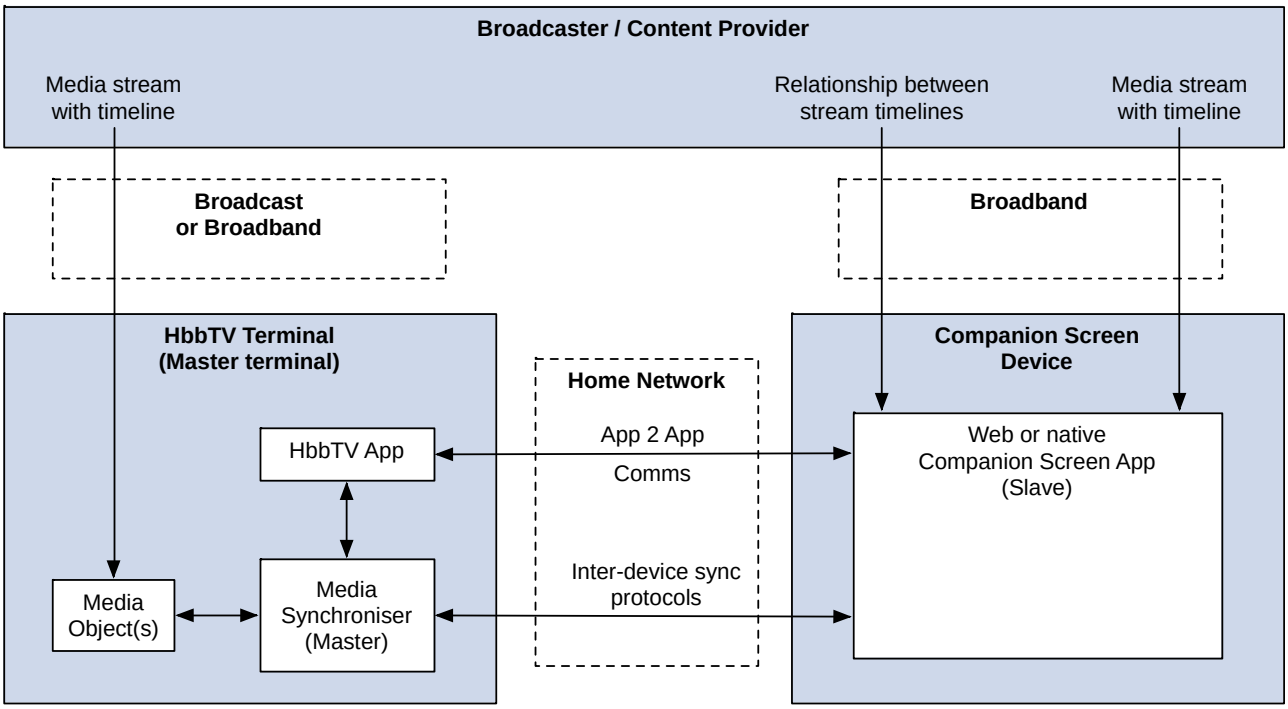


Figure 19: Relationship between MediaSynchroniser object and HbbTV® application for inter-device synchronization with a CSA

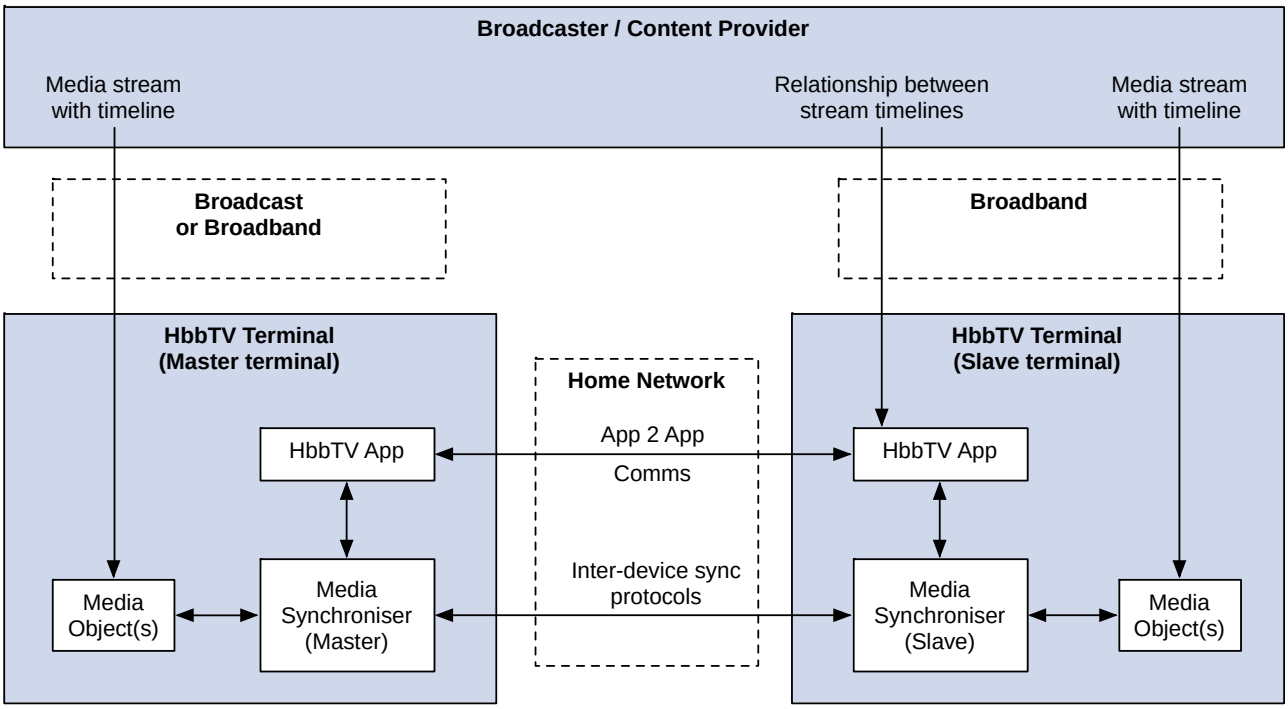


Figure 20: Relationship between MediaSynchroniser object and HbbTV® application for inter-device synchronization with a slave terminal

The master terminal manages the decoding and rendering of the media streams and the master terminal also communicates with the other slave terminal or CSA using the protocols for inter-device synchronization defined in TS 103 286-2 [47].

A terminal becomes a master or slave terminal when inter-device synchronization is enabled on a `MediaSynchroniser` that has been appropriately initialized. The terminal ceases to be a master or slave terminal when inter-device synchronization is disabled, a permanent error occurs during playback or the master media (see clause 13.2.4) stops playing. When an HbbTV® application on a master terminal decides to enable or disable inter-device synchronization is HbbTV® application implementation dependent and outside the scope of the present document.

NOTE: An HbbTV® application can use application to application communication (as defined in clause 14.5) to negotiate when to enable and disable this functionality.

A terminal cannot be a master terminal and a slave terminal simultaneously.

The HbbTV® terminal implements interfaces and protocols defined in TS 103 286-2 [47] under the control of an HbbTV® application through the `MediaSynchroniser` object defined in clause 8.2.3.

Clause 4 of TS 103 286-2 [47] describes an architecture for synchronization that applies to both inter-device synchronization and multi-stream synchronization and defines the concepts of Media Synchronization application Server (MSAS), Synchronization Client (SC), Correlation Information Server (CIS) and Material Resolution Server (MRS). Figure 21 illustrates the how these concepts apply to inter-device synchronization between a master terminal and slave terminal or CSA.

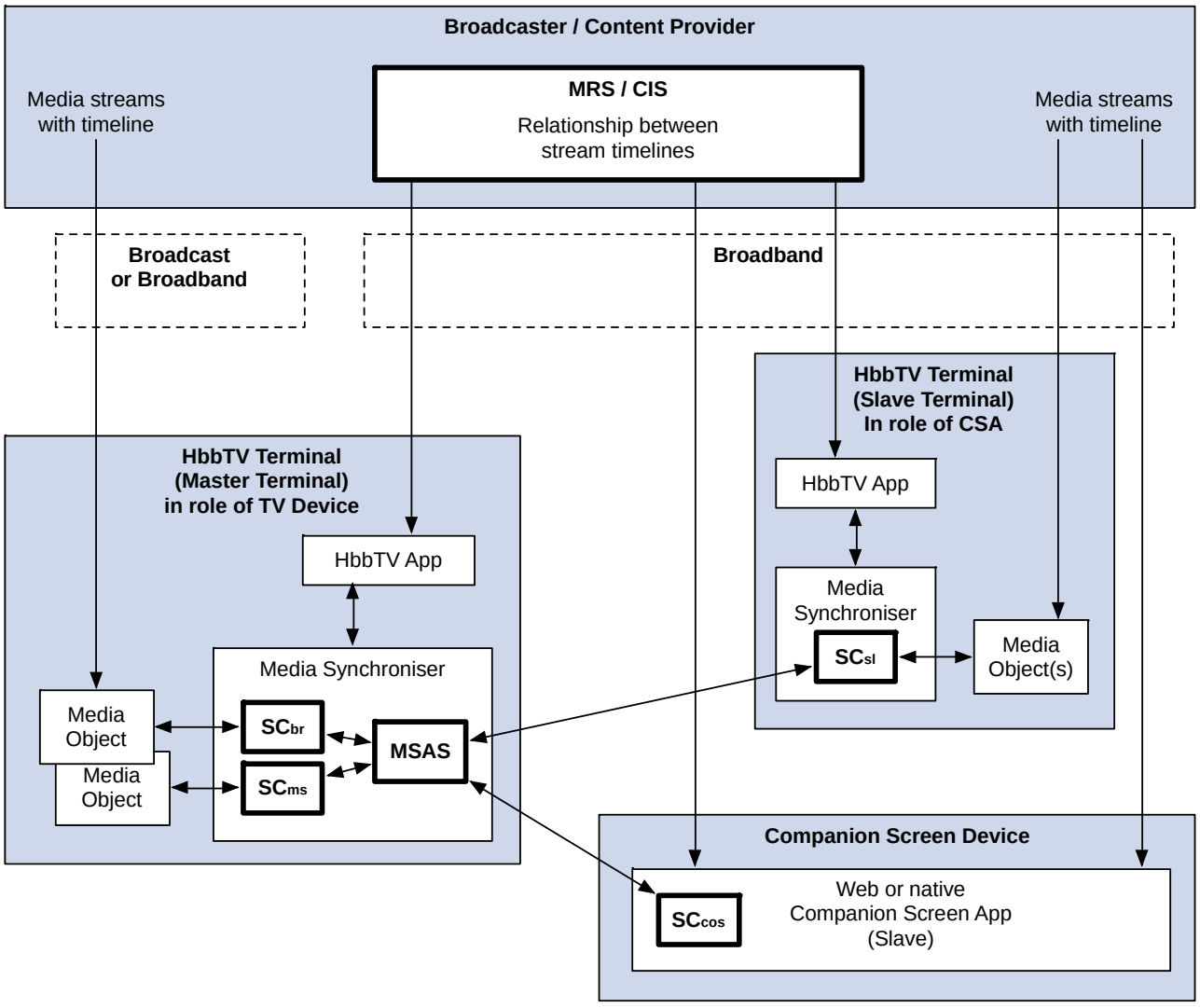


Figure 21: Basic mapping of media synchronization architecture for inter-device synchronization

For an HbbTV® terminal, the `MediaSynchroniser` object performs the inter-device synchronization related elementary functions defined in TS 103 286-2 [47]. This relationship is illustrated in figure 22 for master and slave terminals.

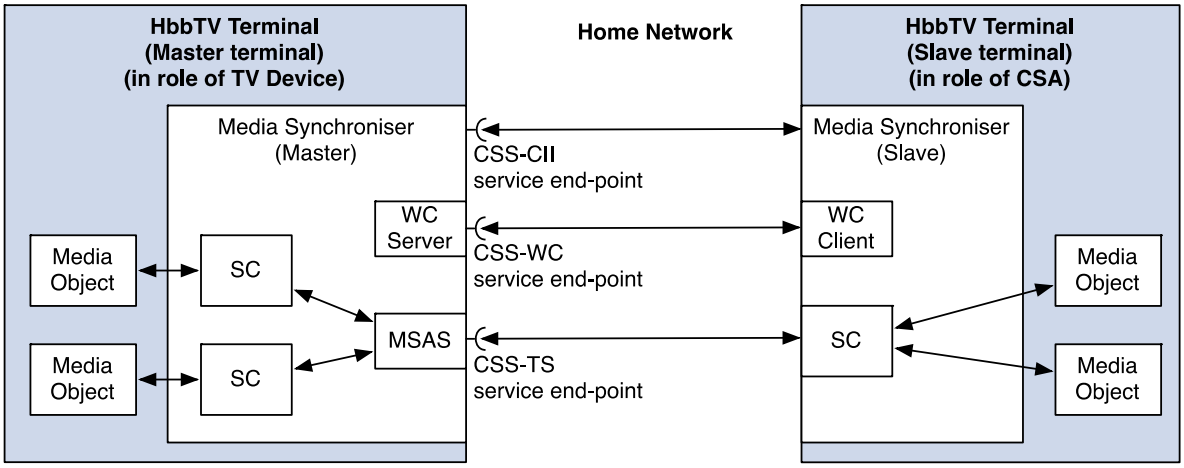


Figure 22: Relationship between the HbbTV® MediaSynchroniser and MSAS, SC, WC Server and WC Client elementary functions and protocol service endpoints

For inter-device synchronization, a terminal and the HbbTV® application running on it is equivalent to either the TV Device or the CSA.

Where a terminal and HbbTV® application acts in the role of a TV Device it is acting as a master terminal. The master terminal contains the MSAS elementary function as well as an SC elementary function (or more than one SC elementary function if simultaneously performing multi-stream synchronization).

A Companion Screen application is a slave. A terminal and HbbTV® application acting in the role of a CSA is also acting as a slave. The slave terminal contains only an SC elementary function.

For each media object being managed by the MediaSynchroniser, the `MediaSynchroniser` performs the role of Synchronization Client to control the timing of presentation of that media object and to communicate information about the timing of presentation to the MSAS function. The MSAS function is implemented by the `MediaSynchroniser` in the master terminal.

In a slave terminal, the `MediaSynchroniser` performs the role of Synchronization Client. If the slave terminal is also performing multi-stream synchronization and is therefore presenting several media objects, then the slave terminal communicates with the master terminal using the inter-device synchronization protocols (defined in TS 103 286-2 [47]) via a single set of protocol connections. The slave terminal does not make a separate set of connections for each media object it is presenting.

The `MediaSynchroniser` functions in the master and slave terminal communicate using the CSS-CII, CSS-WC and CSS-TS protocols defined in TS 103 286-2 [47]. A terminal therefore implements Wall clock server and client functions as well as MSAS and SC functions. The service endpoints for these protocols are provided by the functions of the `MediaSynchroniser` in the master terminal.

13.2.4 Master media and other media

The media object passed as an argument to the `initMediaSynchroniser()` method is the master media being presented by the HbbTV® terminal. Media objects passed as arguments to the `addMediaObject()` method are other media.

From the perspective of a slave terminal, the master terminal is presenting the master media and media objects passed as arguments to the `addMediaObject()` method are other media.

As specified in clause 9.7.1:

- If applications control the presentation timing of the master media using the methods and properties of the corresponding media object then the presentation timing of other media is then adjusted to maintain synchronization with the master media.
- If applications try to control the presentation timing of other media using the methods and properties of the corresponding media object then those actions succeed but the media object is removed and a transient error is generated for the `MediaSynchroniser`.

13.3 Media synchronization states and transitions

13.3.1 States overview (informative)

Multi-stream and inter-device synchronization functionality is controlled via the `MediaSynchroniser` object. Figure 23 shows the states of the terminal and `MediaSynchroniser` object:

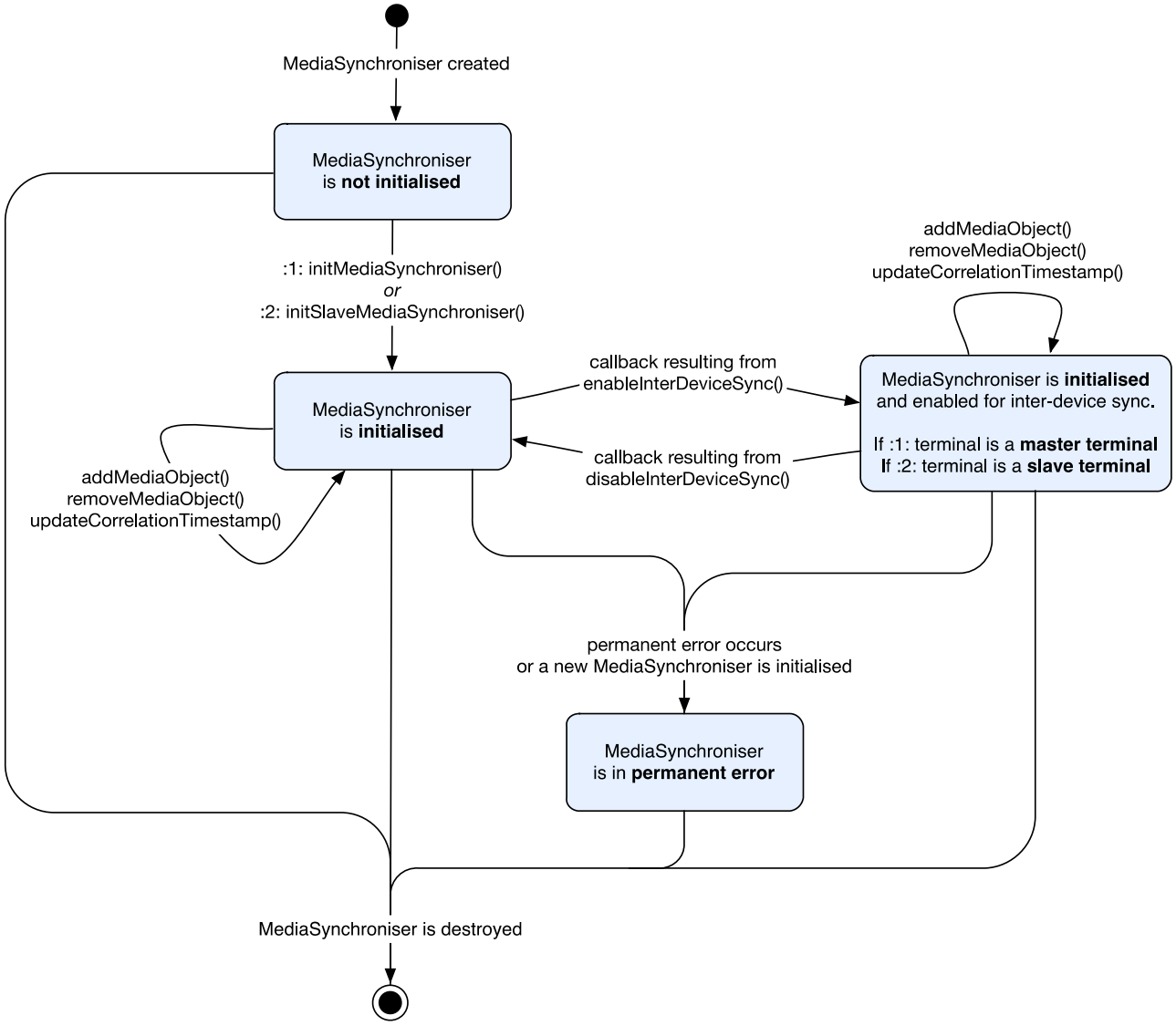


Figure 23: Media Synchronization states

When a `MediaSynchroniser` object is created, it is not yet initialized.

Once a method call to initialize the `MediaSynchroniser` object has completed, the `MediaSynchroniser` object is considered initialized.

While the `MediaSynchroniser` object is initialized, the HbbTV® application can instruct the terminal to add and remove media objects from the `MediaSynchroniser` object and update correlation timestamps (synchronization timing information) for the media objects.

Multi-stream synchronization is performed by the terminal while there are at least two media objects being used with the `MediaSynchroniser` object. These media objects will have been either passed to the `MediaSynchroniser` object during initialization or subsequently added to it.

While the `MediaSynchroniser` object is initialized, the HbbTV® application can enable inter-device synchronization to instruct the terminal to become a master terminal or slave terminal. Whether it becomes a master or slave depends on the method that was used to initialize the `MediaSynchroniser` object. While a terminal is a master terminal or slave terminal it is performing inter-device synchronization.

NOTE: A terminal can only become a slave terminal if the terminal supports this capability (see clause 10.2.9.3).

While the terminal is a master terminal or slave terminal, the HbbTV® application can instruct the terminal to disable inter-device synchronization, causing the terminal to cease to be a master terminal or slave terminal.

A terminal can perform both multi-stream synchronization and inter-device synchronization at the same time by both adding media objects to the `MediaSynchroniser` object and enabling inter-device synchronization.

If a `MediaSynchroniser` object has been previously initialized, but is then replaced by another `MediaSynchroniser` being initialized, then a permanent error occurs for the existing `MediaSynchroniser`.

Permanent errors of the master media stream media object (including the stopping - which is not considered the same as pausing, or the unavailability of the stream's timeline) cause a permanent error of the `MediaSynchroniser`. Permanent errors of other media streams cause those media streams to be removed from the `MediaSynchroniser` and the `MediaSynchroniser` continues operation. Transient errors temporarily suspend synchronization for some or all media objects. See clause 9.7.1.

In any state, if a permanent error occurs then the `MediaSynchroniser` object enters a permanent error state and is no longer initialized and the terminal ceases both multi-stream synchronization and inter-device synchronization (as either master terminal or slave terminal).

If the `MediaSynchroniser` object is destroyed then the terminal ceases both multi-stream synchronization and inter-device synchronization (as either master terminal or slave terminal).

13.3.2 Multi-stream synchronization

While the `MediaSynchroniser` object is initialized, a terminal shall attempt to perform multi-stream synchronization while:

- if the `MediaSynchroniser` was initialized using `initMediaSynchroniser()` method, at least one media object has been added using the `addMediaObject()` method;
- or if the `MediaSynchroniser` was initialized using the `initSlaveMediaSynchroniser()` method, at least two media objects have been added using the `addMediaObject()` method and the inter-device synchronization has been enabled (causing the terminal to have become a slave terminal).

If the combination of media streams is unsupported by the terminal (e.g. a combination not shown as supported in Table 14) then a transient error of the Media Synchroniser object shall occur and the attempt to perform multi-stream synchronisation shall not succeed. The existing presentation of media objects and any currently enabled inter-device synchronisation shall not be interrupted.

The terminal shall cease to perform multi-stream synchronization if:

- the `MediaSynchroniser` was initialized with the `initMediaSynchroniser()` method and all media objects have been removed except for the master media object; or
- the `MediaSynchroniser` was initialized with the `initSlaveMediaSynchroniser()` method and all but one media object has been removed; or
- the `MediaSynchroniser` was initialized using the `initSlaveMediaSynchroniser()` method and inter-device synchronization has been disabled (causing the terminal to cease to be a slave terminal); or

- a permanent error of the `MediaSynchroniser` object occurs (see clause 13.3.8); or
- the `MediaSynchroniser` object is destroyed; or
- another `MediaSynchroniser` object has been initialized after this one was initialized (meaning that the existing `MediaSynchroniser` object has been replaced).

The terminal shall not cease presentation of media objects currently added to the `MediaSynchroniser` object purely as a result of ceasing to perform multi-stream synchronization. However, if there are insufficient decoders available to support all of the media objects, the presentation of one or more media objects may cease due to insufficient resources. In this case, the terminal shall continue to present what was the master media object, taking into account the requirements in clause 10.2.7.

13.3.3 Becoming a master terminal

If inter-device synchronization is enabled for a `MediaSynchroniser` object that was initialized using the `initMediaSynchroniser()` method then the terminal shall ensure that the following protocol endpoints are active:

- a DVB CSS-CII protocol endpoint as described in clause 13.6.2;
- a DVB CSS-WC protocol endpoint as described in clause 13.7.3; and
- a DVB CSS-TS protocol endpoint as described in clause 13.8.2.

Once the endpoints are being provided, the terminal is a master terminal.

NOTE: The application that requested that inter-device synchronization be enabled is notified that the terminal has become a master terminal by callback (see clause 8.2.3.2.2).

13.3.4 Ceasing to be a master terminal

A terminal shall cease to be a master terminal if:

- the `disableInterDeviceSync()` method is called on the `MediaSynchroniser` object (see clause 8.2.3.2.2); or
- there is a permanent error of the `MediaSynchroniser` object (see clause 13.3.8); or
- the `MediaSynchroniser` object is destroyed; or
- another `MediaSynchroniser` object has been initialized after this one was initialized (meaning that the existing `MediaSynchroniser` object has been replaced).

If any of the above is true, then the terminal shall disable inter-device synchronization by disabling the DVB CSS-TS protocol endpoint as described in clause 13.8.2. The terminal may also disable the CSS-CII and CSS-WC protocol endpoints as described in clauses 13.6.2 and 13.7.3.

If the permanent error is due to a state transition for the media object representing the master media that results in the primary aspect of `presentationStatus` changing to "fault" (see clause 13.6.2) then a CII message communicating the `presentationStatus` shall be sent to all slave terminals and CSAs connected to the CSS-CII endpoint. If any endpoints are to be disabled, the CII message shall be sent before this happens.

If the permanent error is due to unavailability of the master media timeline then the terminal shall send a Control Timestamp message to all slave terminals and CSAs connected to the CSS-TS endpoint to communicate that the timeline is not available. If any endpoints are to be disabled, the Control Timestamp message shall be sent before this happens.

When a protocol endpoint is disabled, the terminal shall cleanly close any connections to that endpoint.

Once this process has completed, the terminal is no longer a master terminal.

NOTE: If the application called `disableInterDeviceSync()` method then the application is notified that the terminal is no longer a master terminal by callback (see clause 8.2.3.2.2).

13.3.5 Becoming a slave terminal

If inter-device synchronization is enabled for a `MediaSynchroniser` object that was initialized using the `initSlaveMediaSynchroniser()` method then the terminal shall:

- connect to the master terminal CSS-CII protocol endpoint as described in clause 13.6.3 (if it has not already done so); then
- communicate with the master terminal CSS-WC endpoint as described in clause 13.7.4 (if it is not already doing so); and
- connect to the master terminal CSS-TS protocol endpoint as described in clause 13.8.3.

Once the above steps have occurred then the terminal is a slave terminal and the terminal shall attempt to commence synchronized presentation of any media objects currently added to the `MediaSynchroniser` with the media objects being presented by the master terminal.

NOTE: The application that requested that inter-device synchronization be enabled is notified that the terminal has become a master terminal by callback (see clause 8.2.3.2.2).

13.3.6 Ceasing to be a slave terminal

A terminal shall cease to be a slave terminal if:

- the `disableInterDeviceSync()` method is called on the `MediaSynchroniser` object (see clause 8.2.3.2.2); or
- there is a permanent error of the `MediaSynchroniser` object (e.g. because the master terminal ceases to be a master terminal, or the timeline is no longer available, or there is some problem with communicating with the protocol endpoints) (see clause 13.3.8); or
- the `MediaSynchroniser` object is destroyed; or
- another `MediaSynchroniser` object has been initialized after this one was initialized (meaning that the existing `MediaSynchroniser` object has been replaced).

If any of the above is true, the terminal shall:

- disconnect any connections to the DVB CSS-CII and DVB CSS-TS protocol endpoints of the master terminal; and
- cease synchronization of all media objects associated with the `MediaSynchroniser`.

The terminal may also cease to communicate with the DVB CSS-WC protocol endpoint of the master terminal.

Once this process has completed the terminal is no longer a slave terminal.

The terminal shall not cease presentation of media objects currently added to the `MediaSynchroniser` as a result of ceasing to be a slave terminal.

13.3.7 Transient errors

When a transient error of the `MediaSynchroniser` occurs the `MediaSynchroniser` shall continue to attempt to perform multi-stream and/or inter-device synchronization if it is already doing so.

The following situations shall cause a transient error of the `MediaSynchroniser`:

- the combination of media streams being requested for multi-stream synchronisation (due to a call to the `addMediaObject()` method) being unsupported (see clause 13.3.2)
- errors of media objects representing other media streams (see clause 9.7.1);
- unavailability of the requested timeline for other media (see clause 9.7.3);
- inability to achieve or maintain synchronization between streams because the content data of the streams cannot be obtained early enough or delayed sufficiently (see clauses 9.7.2 and 13.8.3.5);

- buffering, stalling or transient errors of the master media stream or other media streams (see clause 9.7.1);
- calling of methods of the `MediaSynchroniser` when the `MediaSynchroniser` is in an inappropriate state (see clause 8.2.3.2.2);
- other media streams not being in a suitable state to participate in synchronization (see clause 9.7.1); or
- the `presentationStatus` received by a slave terminal from a master terminal in a CII message has changed to "transitioning" (see clause 13.6.3).

NOTE: These causes of transient errors can occur at any time while inter-device or multi-stream synchronization is being performed. They do not just occur as a result of method calls by the application.

13.3.8 Permanent errors

When a permanent error of the `MediaSynchroniser` occurs, the `MediaSynchroniser` shall enter the permanent error state. As specified in clauses 13.3.2, 13.3.4, and 13.3.6, this shall cause the terminal to cease to perform multi-stream synchronization if it is currently performing it. If the terminal is a master terminal or slave terminal then the terminal shall also cease to be a master terminal or slave terminal.

When in the permanent error state, all method calls on the `MediaSynchroniser` object shall fail.

NOTE 1: To perform further media synchronization, the existing `MediaSynchroniser` object needs to be destroyed or discarded and a new one created and initialized.

A permanent error of the `MediaSynchroniser` can occur if any of the following occurs:

- errors during initialization of the `MediaSynchroniser` (see `initMediaSynchroniser()` and `initSlaveMediaSynchroniser()` methods in clause 8.2.3.2.2);
- errors of media objects representing the master media (see clause 9.7.1);
- unavailability of the timeline for the master media (see clause 9.7.3);
- errors in communication with a master terminal using the protocols defined in TS 103 286-2 [47]. when acting as a slave terminal (see clauses 13.6.3 and 13.8.3);
- the master media stream not being in a suitable state to participate in synchronization, including the media stream transitioning to a stopped, unrealised or finished state or as a consequence of the media source being re-loaded (see clause 9.7.1);
- the `MediaSynchroniser` being replaced after it has been initialized because another `MediaSynchroniser` has subsequently been initialized (see `initMediaSynchroniser()` and `initSlaveMediaSynchroniser()` methods in clause 8.2.3.2.2).

NOTE 2: With the exception of the first cause listed above, these causes of permanent errors can occur at any time while inter-device of multi-stream synchronization is being performed. They do not only occur as an immediate effect of method calls by the application.

13.4 Timelines and timestamping

13.4.1 Reference point for timestamping

The reference point for generation and interpretation of Timestamps by the HbbTV® terminal and Companion Screen application used in inter-device synchronization is as defined in clause 5.7.2 of TS 103 286-2 [47].

The terminal is responsible for compensating for any extra travel time behind a technically implemented timestamp measurement point due to output buffers, frame buffers, quality-enhancement technologies and other sources of delay between the point of measurement and the emission of sound or light. Clause C.4 of TS 103 286-2 [47] provides examples of such calculations. If no accurate values are known then the terminal shall make a best-effort estimate of the extra travel time and compensate for it.

NOTE: HDMI 2.0 [i.7] provides functionality for dynamic synchronization of video and audio streams. Information from the HDMI can be used to make a best-effort estimate of the extra travel time between a set-top box and the light and sound output of the TV screen.

13.4.2 Supported timelines and their selection

A Timeline is the reference frame for measuring the progress of time for a given media stream. How a timeline is to be derived for a given media stream is described by a Timeline Selector as defined in clause 5.3 of TS 103 286-2 [47].

For multi-stream and inter-device synchronization, the terminal shall support the use of the following types of timeline (defined in this clause and clause 5.3 of TS 103 286-2 [47]) for the types of broadcast or broadband content shown in Table 18.

Table 18: Multi-stream and inter-device synchronization timeline support

Type of timeline	Supported for
MPEG-TS Presentation Timestamps (PTS) (see note 1)	MPEG Transport Stream delivered via broadcast (see note 2). Single program MPEG Transport Stream streamed via broadband.
ISOBMFF Composition Time (CT) (see note 1)	ISOBMFF streamed using HTTP via broadband (excluding MPEG DASH).
MPEG-TS Timed External Media Information (TEMI) (see note 1)	MPEG Transport Stream delivered via broadcast (see note 2). Single program MPEG Transport Stream streamed via broadband.
MPEG DASH Period Relative (see note 1)	MPEG DASH streamed via broadband.
EBU-TT-D milliseconds	EBU-TT-D conformant document delivered via HTTP.
NOTE 1: This type of timeline is defined in clause 5.3 of TS 103 286-2 [47]. NOTE 2: This includes MPEG Transport streams originally delivered by broadcast and then played back from a PVR recording, provided that the component carrying the timeline was recorded. Timelines not present on a component that was recorded are not available when the recording is played back.	

The terminal reports which timeline types are supported for different types of broadband streaming in the XML capabilities document defined in clause 10.2.4 by listing supported timeline types in the value of the `sync_tl` attribute of `<video_profile>` elements (see clause A.2.14). For a media stream that is a raw XML document containing EBU-TT-D conformant TTML subtitles that is delivered out of band the terminal shall support the use of a timeline derived from the timing information used within the EBU-TT-D conformant TTML subtitle document. The Timeline Selector for this timeline shall be "`urn:hbbtv:sync:timeline:ebu-tt-d`". An EBU-TT-D conformant TTML subtitle timing is expressed in terms of hours, minutes and seconds, where the seconds can include a decimal fractional part.
1 millisecond shall equal 1 tick and therefore the equivalent ticks time value on this Timeline shall be calculated as follows, rounding to the nearest integer:

$$\text{round}\ (1\,000 \times (\text{seconds} + 60 \times (\text{minutes} + 60 \times \text{hours})))$$

NOTE 1: EBU-TT-D conformant TTML subtitles delivered out-of-band can also be synchronized with other media streams using extensions to the A/V Control object as defined in A.2.5.3 and HTML5 media objects as defined in clause A.2.12.2.

The terminal shall derive the timeline described by a Timeline Selector for a media stream if the Timeline Selector and media object representing that media stream are passed as arguments to the `initMediaSynchroniser()` or `addMediaObject()` methods of a `MediaSynchroniser` object.

TS 103 286-2 [47] defines support in the terminal for the decoding of MPEG-TS Timed External Media Information (TEMI) timeline descriptors in the adaptation field of Transport Stream packets carrying Packetized Elementary Streams (PES). Terminals shall support at least the following components of a DVB service to carry MPEG TEMI timeline descriptors:

- Any component that is supported by the terminal for use with media synchronization and MPEG TEMI, i.e. audio, video and subtitles.
- Any component with stream_type 6 (private PES) and stream_id 1011 1101 ("private_stream_1") in the PES packet header, including, but not limited to, components where the PES packet payloads are empty.

NOTE 2: Selection of the correct timeline descriptors by component tag and timeline id is done via the timeline selector by using the media sync API as defined in clause 8.2.3. This also means that there can be different timelines present if applications use either multiple components or timeline ids or a combination of both.

The terminal shall support decoding a minimum of 2 different TEMI timelines simultaneously where each can be carried in a different component. If a terminal has insufficient resources to decode a requested TEMI timeline, then the terminal behaviour is as if the requested timeline is unavailable (see clause 13.8.2.2).

EXAMPLE: An HbbTV application initialises a MediaSynchroniser, using broadcast audio and video and a timeline selector that specifies a TEMI timeline on component tag 1 with timeline_id of 0x80. The application also enables inter-device synchronisation in the role of a master terminal. This enables a CSA to synchronise to the terminal, requesting a different TEMI timeline that is carried on component tag 2 with timeline_id of 0x05.

When deriving a timeline from TEMI timeline descriptors, a wrap of PTS values shall not affect the TEMI timeline.

NOTE 3: A broadcaster can choose to not include a temi_timeline_descriptors for every access unit. Extrapolation of the timeline position for access units without a temi_timeline_descriptor (following an earlier access unit that did have a temi_timeline_descriptor) is involves calculating the difference between the PTS of the two access units. This calculation needs to correctly handle a situation where PTS has wrapped between the two access units.

13.4.3 Synchronization timeline

13.4.3.1 Timelines for the MediaSynchroniser API

The Timeline to be used by the `MediaSynchroniser` API within a master terminal is the timeline selected for the master media in the call to the `initMediaSynchroniser()` method of the `MediaSynchroniser` object.

For a slave terminal, the Timeline to be used by the `MediaSynchroniser` API is the same Timeline that is used by the `MediaSynchroniser` API at the master terminal.

Correlation Timestamps provided by HbbTV® applications to the `MediaSynchroniser` (see clause 8.2.3.4) are therefore interpreted by both master and slave terminals as follows:

- `tlvMaster` value represents a point on the Timeline used by the `MediaSynchroniser` API;
- `tlvOther` value represents a point on the timeline selected for the other media that this Correlation Timestamp is associated with.

The timeline for the other media is specified when the media object representing it is added to a `MediaSynchroniser` using the `addMediaObject()` method.

13.4.3.2 Synchronization timeline for Inter-device synchronization

For inter-device synchronization, the Synchronization Timeline is the reference frame for `contentTime` values in timestamp messages exchanged between a master terminal and a slave terminal (or CSA) via the CSS-TS protocol (as described in clauses 5.7 and 9 of TS 103 286-2 [47]).

Where timestamps are being exchanged between a master terminal and a slave terminal via the CSS-TS protocol, the Synchronization Timeline is the same as the Timeline for the `MediaSynchroniser` API used by both the master and slave terminal.

NOTE 1: In the context of the CSS-TS protocol (see clause 13.8.3.2), the slave terminal will select the timeline that was passed to it by the master terminal via the CSS-CII protocol (see clause 13.6.3). The timeline that is passed is the timeline used by the `MediaSynchroniser` API at the master terminal (see clause 13.6.2).

However, where timestamps are exchanged between a master terminal and a CSA, the Synchronization Timeline may be a different timeline.

NOTE 2: In the context of the CSS-TS protocol, a CSA is not required to select the timeline that was passed to it by the master terminal via the CSS-CII message.

To use a TEMI timeline, the terminal shall decode the `temi_timeline_descriptor` with `timeline_id` matching that specified in the timeline selector (as specified in clause 11.3.3 of TS 103 286-2 [47]) when carried in an `af_descriptor` within the adaptation header of MPEG transport stream packets corresponding to the component specified in the timeline selector. The terminal is not required to decode other descriptors that can be carried in the `af_descriptor`. Decoding of TEMI timelines, as specified by the timeline selector, shall be supported for all values of the `timeline_id` between 0x00 and 0xff inclusive and independent of the presence of any `temi_location_descriptor` with the same `timeline_id`.

13.5 Buffer for media synchronization

13.5.1 General

Terminals have buffers for normal operation, for example input and output buffers for codecs. In the case of media synchronization between two or more pieces of timed content, additional buffer capacity is needed, as one timed content will be the most laggard and the other piece(s) of timed content need to be buffered to achieve time alignment. That additional buffer capacity can be in the terminal and/or in the network.

- The buffer for media synchronization in the terminal is a reserved amount of memory dedicated to media synchronization, which is additional to the existing buffering. An HbbTV® application can use this to buffer a timed content in order to synchronize it with another timed content.
- Buffering for media synchronization can also be performed in the network. A Content Delivery Network (CDN) can cache live and on-demand content. The terminal instructs the retrieval of chunks of the timed content from the network such that the terminal can play out the timed content with the correct timing for media synchronization without running out of the buffer space that the terminal has for its normal operation.

Clause 13.5.2 defines the cases for the use of the different buffers, or the absence thereof.

Clause 13.5.3 defines the buffer model for the buffer for media synchronization in the terminal.

The clauses of 13.5 apply to both multi-stream and inter-device synchronization.

13.5.2 Media synchronization buffering cases

For any timed content one of the cases of Table 19 applies.

Table 19: Buffering for media synchronization of timed content in the network

Case	The timed content was PVR-recorded	The timed content is buffered in the network	The terminal has a buffer for media synchronization
1	Yes	N/A	N/A
2	No	Yes	N/A
3	No	No	Yes
4	No	No	No

In case 1, the timed content was recorded on the PVR in the past. In this case, obviously no additional buffer capacity for media synchronization (network or terminal) is needed for that timed content.

In case 2, the terminal shall use the media synchronization buffer in the network for the time alignment of the timed content and it shall not reduce the available capacity of the buffer in the terminal for media synchronization (clause 13.5.3) for other timed content. It is the broadcaster's responsibility to assure that the relevant chunks of timed content are available from the network. An application can determine whether an item of timed content can be buffered in the network by inspecting the source and type of the timed content. Table 20 defines whether timed content can be buffered in the network or not.

Table 20: Media Synchronization buffering of timed content in the network

Source	Type	Is buffered in the network
Broadcast	any	No
Broadband	HTTP Streaming	No
	MPEG DASH	Yes

In case 3, the terminal shall use its buffer for media synchronization to time the playout of the timed content with the correct timing for media synchronization. The media synchronization buffer model is provided in clause 13.5.3. It is the broadcaster's responsibility to assure that the terminal will not need more than the available buffer size for media synchronization.

In case 4, media synchronization is still possible. It is the broadcaster's responsibility to assure that there is not more than one timed content of this type in a media synchronization session, and that it will always be the most laggard timed content, so that it does not need to be buffered for media synchronization.

The broadcaster may need to employ delay management in its distribution networks to achieve coarse time alignment between the received media streams, in order to prevent buffer overflow or underflow in the terminal, network or Companion Screen Device.

NOTE: Informative clause G.2 provides implementation guidelines for media synchronization for managing delay throughout distribution network. More implementation guidelines for broadcasters are provided in annex B of TS 103 286-2 [47].

13.5.3 Media synchronization buffer model

The media synchronization buffer is optional. The terminal exposes the presence of a buffer for media synchronization through a non-zero value of the property `minSyncBufferSize` of the `MediaSynchroniser` embedded object (see clause 8.2.3.2.1).

The following are requirements on media synchronization buffer model and size.

- The terminal shall conceptually have a single dedicated shared buffer for all types of media synchronization together.
  - Multi-stream synchronization: synchronising multiple media streams on a single HbbTV® terminal, both broadcast streams and companion streams.
  - Inter-device synchronization: synchronizing media streams between an HbbTV® terminal and a companion device or other HbbTV® terminal.
  - Synchronising application and content across devices.
- The media synchronization buffer in the terminal shall only be used for buffering content to be presented on the HbbTV® terminal itself.
- The terminal shall, for media synchronization, buffer all service components that are used for the synchronization, if the service is part of a multiplex.
- When a service is buffered for media synchronization, the timing of any events that are part of the service shall be preserved. For example, do-it-now stream events included in an MPEG-2 transport stream shall be fired at the same time as the presentation of the video and audio that they would have been fired with if the service had not been buffered for media synchronization. Events that are part of the service shall not be fired when the part of the stream carrying them is first buffered.
- The terminal may also buffer components from other services, when available media synchronization buffer space permits. If components of other services are buffered then it does not count towards the usage of the media synchronization buffer.
- The terminal shall manage the media synchronization buffer in way such that in case of a resource conflict, data for components of the services that are currently used for synchronization on the terminal, ousts any other data from the media synchronization buffer.

NOTE 1: This means that, if an implementation decides to store non-service-components like the multiplexing overhead or the full multiplex, then the defined buffer space is not used for this overhead.

- The size of the media synchronization buffer shall be at least 30 Megabytes (that is 30 × 1 024 × 1 024 = 31 457 280 bytes).

NOTE 2: As a consequence of the above requirement, the value of the property minSyncBufferSize (clause 8.2.3.2.1) is either 0 (i.e. no minimum guaranteed buffer space for media synchronization), or a number equal to or greater than 31 457 280.

NOTE 3: Calculation example. If the broadcast stream has a bit rate of 20 Mb/s and the media synchronization buffer size is 30 MB, then the broadcaster can assume that the terminal is able to buffer up to 30 (Megabyte) × 8 (bit per byte) / 20 (Megabit per second) = 12,58 seconds of media- stream buffering. If there are two media streams to be buffered (e.g. in a combination of multi-stream and inter-device synchronization), then there would be at average 6 seconds of media synchronization buffering per media stream.

Whereas specification of media synchronization buffer size of the Companion Device is out of scope of the present document, it may be reasonably expected to be similarly big as for the terminal.

13.6 Content Identification Information service endpoint

13.6.1 General

To facilitate inter-device synchronization of the presentation of media, a master terminal implements the CSS-CII service endpoint (as defined in clause 6 of TS 103 286-2 [47]). A Companion Screen application or a slave terminal subscribes to the CSS-CII service endpoint when performing inter-device synchronization.

13.6.2 CSS-CII service endpoint (master terminal)

A master terminal shall implement a CSS-CII service endpoint as defined in clause 6 of TS 103 286-2 [47] at the terminal's broadband interface.

The master terminal shall provide an active CSS-CII protocol service endpoint when the HbbTV® application has enabled inter-device synchronization functionality. The terminal may provide an active CSS-CII protocol service endpoint at other times but this is implementation dependent and outside the scope of the present document. The master terminal shall support a minimum of 5 concurrent connections to the CSS-CII service endpoint and shall allow slave terminals and CSAs to connect to this service endpoint until the master terminal has reached the limit of the number of simultaneous sessions it can support.

The master terminal shall ignore the `Origin` header if it is present in the client handshake of the connection request.

CII messages sent by the master terminal via a connection to the CSS-CII service endpoint shall convey the following:

- The `contentId` and `contentIdStatus` properties shall correspond to the Content Identifier of the master media. For DVB broadcast services (and PVR recordings made from them) and MPEG DASH streams this shall be as defined in clause 5.2 of TS 103 286-2 [47]. For ISOBMFF and MPEG2 TS delivered via broadband:

  - the value of the `contentId` property shall be the absolute version of the URL provided by the HbbTV® application to specify the location of the media stream, before any redirect that may occur, and

  - the `contentIdStatus` shall be "`final`".

NOTE 1: When playing back a PVR recording of a DVB broadcast service, the contentId represents the original broadcast. Although the contentId incorporates elements that come from components that are not necessarily recorded (e.g. NIT, BAT and SDT) these elements are considered pseudo static and therefore can be captured once during the recording process for inclusion in the contentId during playback.

- The `presentationStatus` property shall describe the presentation status of the master media. The primary aspect of presentation status shall be derived from the state of the media object presenting the master media. For a video/broadcast object this shall be according to Table 21. For an A/V Control object this shall be according to Table 22. For an HTML5 media element that shall be according to Table 23.

NOTE 2: While the master media is paused, buffering, tuning or presenting normally, the primary aspect of status is expected to be "okay" or "transitioning" as appropriate (see clause 5.6.4 of TS 103 286-2 [47]). If there are temporary disruptions to picture and sound (e.g. due to poor broadcast signal reception) but the media continues to be presented without generating a permanent error condition, then the primary aspect of presentationStatus remains "okay" because presentation is continuing.

- The `mrsUrl` property shall correspond to the URL of the MRS determined for the master media (see clause 5.6.2 of TS 103 286-2 [47] for MPEG TS delivered via broadcast and for MPEG DASH).

NOTE 3: No mechanism is defined to determine the MRS URL if the master media is MPEG2 TS delivered via broadband (not via broadcast or DVB IPTV) or if it is ISOBMFF (not DASH) delivered via broadband. In these circumstances the value of the `mrsUrl` property is null because the terminal cannot provide an MRS URL.

- The `wcUrl` property shall correspond to the CSS-WC service endpoint provided by the master terminal (see clause 13.7).

- The `tsUrl` property shall correspond to the CSS-TS service endpoint provided by the master terminal (see clause 13.8).

- While the `MediaSynchroniser` API timeline is available (see clause 9.7.3) the timelines property shall convey a list where the first item in the list is a timeline options JSON object (as defined in clause 5.6 of TS 103 286-2 [47]) that describes the `MediaSynchroniser` API Timeline (as defined in clause 13.4.3).

Table 21: Primary aspect of presentationStatus when master media is a video/broadcast object

Transition to this state of the v/b object	Current v/b object state	Primary aspect of CSS-CII presentationStatus
channel change or bind	Connecting	transitioning
transient error	Connecting	okay
any transition	Presenting	okay
any transition	Unrealised or Stopped	fault see note
NOTE: After this is sent in a CII message, the terminal will also cease to be a master terminal for inter-device synchronization because this scenario generates a permanent error (see clause 9.7.1).		

Table 22: Primary aspect of presentationStatus when master media is an A/V Control object

States of the A/V Control object since data attribute last changed	Current A/V Control object state	Primary aspect of CSS-CII presentationStatus
Has not yet been in Playing state	Connecting, Buffering or Paused	transitioning
Has been in Playing state	Buffering or Paused	okay
	Playing	okay
	Error, Stopped or Finished	fault see note
NOTE: After this is sent in a CII message, the terminal will also cease to be a master terminal for inter-device synchronization because this scenario generates a permanent error (see clause 9.7.1).		

Table 23: Primary aspect of presentationStatus when master media is an HTML5 media element

State of HTML5 media element	Primary aspect of CSS-CII presentationStatus
readyState < HAVE_CURRENT_DATA	transitioning
readyState ≥ HAVE_CURRENT_DATA	okay
An error has occurred	fault see note
NOTE: After this is sent in a CII message, the terminal will also cease to be a master terminal for inter-device synchronization because this scenario generates a permanent error (see clause 9.7.1) .	

As described in clause 5.6 of TS 103 286-2 [47], a slave terminal (or CSA) assumes initial values for all properties of null until a first CII message is received from the master terminal. An active CSS-CII service endpoint is always accompanied by active CSS-WC and CSS-TS service endpoints and there is always a designated master media object that will be connecting to or playing a source of media. The first CII message shall therefore define non-null values for at least the following properties: `protocolVersion`, `contentId`, `contentIdStatus`, `presentationStatus`, `tsUrl` and `wcUrl`.

NOTE 4: Because the `timelines` property is to be defined if the MediaSynchroniser API timeline is available, then the first CII message is also expected to include this property if the timeline is available at the time.

NOTE 5: The `timelines` property can also list other timelines provided that the MediaSynchroniser API timeline is the first item in the list.

As described in clause 5.6 of TS 103 286-2 [47], properties may be omitted from CII messages if the value of the property has not changed since the last time a CII message was sent to the same slave.

NOTE 6: The contentId and other properties can change during inter-device synchronization even though the master media is derived from the same media object. The master terminal pushes updates when any property of the CII message changes.

The CSS-CII endpoint shall satisfy the security requirements of clause 11.7 of the present document.

13.6.3 Use of CSS-CII service endpoint (slave terminal)

When a terminal becomes a slave terminal, it shall connect to the CSS-CII service endpoint of the master terminal as indicated by the URL of the service endpoint provided as an argument to the `initSlaveMediaSynchroniser()` method if it has not already done so.

If any of the following situations occur, then the slave terminal shall deem that inter-device synchronization with the master terminal has failed and generate a permanent error with the corresponding error code (defined in clause 8.2.3.2.4) for the `MediaSynchroniser` object and cease to be a slave terminal:

- The master terminal refuses the request to connect to the CSS-CII service endpoint (error code 6).
- The CSS-CII service endpoint is, or becomes, unreachable or unresponsive (error code 6).
- The master terminal closes the connection (error code 6).
- The `wcUrl` property of any CSS-CII message received is null or is not provided in the first CSS-CII message received (error code 10).
- The `tsUrl` property of any CSS-CII message received is null or is not provided in the first CSS-CII message received (error code 10).
- The `presentationStatus` property of any CSS-CII message received has a value where the primary status aspect is not "okay" or "transitioning" (error code 6).

NOTE: A CSS-CII message sent to a slave terminal can omit properties if they have not changed since the previous message sent.

If none of the above situations occurs, the slave terminal shall maintain the connection to the CSS-CII service endpoint until the terminal ceases to be a slave terminal (see clause 13.6.5) or the `MediaSynchroniser` in the slave terminal encounters a permanent error.

If the `presentationStatus` property changes to "transitioning" then the terminal shall generate a transient error for the `MediaSynchroniser` object with error code 19.

13.7 Wall clock synchronization

13.7.1 General

To facilitate inter-device synchronization of the presentation of media, a terminal has an internal Wall Clock against which the progress of the timeline of media being presented by the terminal can be measured. The master terminal responds to Wall Clock Synchronization protocol requests from slave HbbTV® terminals or a Companion Screen applications to synchronise their own internal Wall Clock with that of the master terminal.

13.7.2 Wall clock properties

The terminal shall have a Wall Clock as defined in clause 8.3 of TS 103 286-2 [47]. The Wall Clock of the master terminal shall be monotonic and without discontinuities. The Wall Clock should not be directly derived from any real time clock source in the master or slave terminal.

NOTE 1: It is possible to derive a Wall Clock from a real time clock source, but this requires care to be taken to ensure that it is free from discontinuities and meets the measurement precision and maximum frequency error requirements described below. A local NTP client process within the terminal can cause discontinuities or contribute to frequency error when it is applying a frequency adjustment (slew) to adjust the clock.

Measurements of the Wall Clock (or clock from which the Wall Clock is derived) by the terminal shall have a measurement precision (as defined in clause 8.2.2 of TS 103 286-2 [47]) of 1ms or better (smaller) for the purposes of:

- the master or slave terminal measuring the timeline of the broadcast or broadband media against the reference point (defined in clause 13.2.5);
- the master terminal setting the value of the `receive_timevalue` in a Wall Clock Synchronization response message with `message_type` 1, 2 or 3;
- the master terminal setting the value of the `transmit_timevalue` in a Wall Clock Synchronization response message with `message_type` 1 or 3;
- the slave terminal setting the value of the `originate_timevalue` in a Wall Clock Synchronization request message with `message_type` 0; and
- the slave terminal recording the time at which at which a Wall Clock Synchronization response message with `message_type` 1, 2 or 3 is received.

NOTE 2: A master terminal sets the `precision` field in Wall Clock Synchronization response messages that it sends in order to indicate the measurement precision for `receive_timevalue` and `transmit_timevalue` fields. The `precision` field is not used for this purpose in Wall Clock Synchronization request messages sent by slave terminals.

The maximum frequency error of the Wall Clock (or clock from which the Wall Clock is derived), as defined in clause 8.2.3 of TS 103 286-2 [47], shall be 500 ppm or better (smaller).

NOTE 3: A master terminal sets the `max_freq_error` field in Wall Clock Synchronization response messages that it sends in order to indicate the maximum frequency error of the Wall Clock. The `max_freq_error` field is not used for this purpose in Wall Clock Synchronization request messages sent by slave terminals.

13.7.3 WC-Server (master terminal)

When the terminal is a master terminal, it shall implement a WC-Server as defined in clause 8 of TS 103 286-2 [47]. The WC-Server shall provide the CSS-WC service endpoint on the terminal's broadband interface.

The master terminal WC-Server function shall provide an active Wall Clock Synchronization service endpoint and advertise the location through the `wcUrl` property of CSS-CII messages sent from the CSS-CII service endpoint. The terminal shall not change the location of the WC-Server endpoint while one or more clients are connected to CSS-CII.

NOTE 1: A slave terminal or a CSA only communicates with CSS-WC service endpoint on a master terminal. A terminal is not required to implement a WC-Server function while it is a slave terminal.

The WC-Server shall respond in a timely fashion to a minimum of 25 requests per second. Responding in a timely fashion is defined as sending all responses within 200 ms or less of receiving any request, given uncongested network conditions on the terminal's broadband interface.

NOTE 2: 25 requests per second is assumed to comprise 5 entities (some combination of CSAs or slave terminals) simultaneously sending 5 requests per second. This is a peak rate that may be used only for a few seconds at the beginning of a Wall Clock Synchronization procedure to rapidly synchronise their Wall Clocks. Subsequent requests can be assumed to be much more infrequent (e.g. 1 every 2 seconds per entity).

If the WC-Server responds to a request by sending both a response and a follow-up response then the follow-up response shall also be sent by the terminal within 200 ms of the request being received, given uncongested network conditions on the terminal's broadband interface.

In Wall Clock response messages (where the `message_type` field has value 1, 2 or 3) sent by a master terminal the `precision` field shall have a value equal to or less than -9 and the `max_freq_error` field shall have a value equal to or less than 12 800.

NOTE 3: These constraints on values correspond to the requirements specified in clause 13.7.2 for measurement precision (for setting the values of the `receive_timevalue` and `transmit_timevalue` fields) and maximum frequency error of the Wall Clock.

13.7.4 WC-Client (slave terminal)

While a terminal is a slave terminal, or in the process of becoming a slave terminal, it shall implement a WC-Client function (as defined in clause 8 of TS 103 286-2 [47]).

During the process of becoming a slave terminal, the WC-Client function of the slave terminal commences the process of Wall Clock Synchronization if it has not already done so. The WC-Client function shall send a Wall Clock protocol request message within 5 seconds of both becoming a slave terminal and having obtained the endpoint location for the CSS-WC interface via the CSS-CII interface. When the terminal ceases to be a slave terminal, the WC-Client function of the slave terminal may cease this process.

The WC-Client function of the slave terminal shall send Wall Clock protocol request messages to the service endpoint of the WC-Server function that is located by the `wcUrl` property in CSS-CII messages received from the master terminal.

The WC-Client function of the slave terminal shall wait at least 200 ms and no more than 5 000 ms between sending request messages. A WC-Client function shall send no more than 30 requests over a 60 second period, ignoring the first 2 seconds after the Wall Clock Synchronization process is commenced.

The WC-Client function of the slave terminal shall be able to synchronise its Wall Clock when response or follow-up response messages arrive at the WC-Client up to at least 1 000 ms after the request message is sent by the WC-Client function that caused the response or follow-up response message to be sent.

If the WC-Client function of the slave terminal does not receive an expected response or follow-up response message within 1 000 ms, the WC-Client function shall continue to send request messages and use the received response and follow-up response messages for synchronising its Wall Clock.The algorithm by which the WC-Client function of the slave terminal estimates and adjusts the slave terminal Wall Clock to match the Wall Clock of the master terminal is implementation dependent and outside the scope of the present document.

NOTE 1: Clause C.8.3 of TS 103 286-2 [47] provides guidance on how a WC-Client estimates the master terminal Wall Clock.

Dispersion is the maximum amount by which the WC-Client estimate of the master WC-Server Wall Clock could differ from the true value of the WC-Server Wall Clock. The peak value of dispersion over an interval of time quantifies the limit on the accuracy of synchronization achieved by the slave terminal during that interval. The slave terminal reports this as a positive number of milliseconds through the `interDeviceSyncDispersion` property of the `MediaSynchroniser` object (see clause 8.2.3.2.1). The value of this property shall represent the peak value of dispersion over the period since the last time the property was updated.

The algorithm by which dispersion is calculated is implementation dependent and outside the scope of the present document.

NOTE 2: The main contribution to dispersion is the round trip time of the request and response messages on the home network. Between receiving responses, dispersion increases, reflecting the potential for slight frequency differences between the oscillators driving the Wall Clock in each device. Clause C.8.3 of TS 103 286-2 [47] also provides guidance on how to calculate dispersion.

Before the WC-Client function of the slave terminal receives its first response from a WC-Server, the dispersion shall be infinite.

NOTE 3: Any suitable algorithm for adjusting the slave terminal Wall Clock will lower dispersion whenever an adjustment is made. Peak values for calculated dispersion therefore occur at the moments immediately prior to the adjustment being made.

13.8 Timeline Synchronization service endpoint

13.8.1 General

To facilitate inter-device synchronization of the presentation of media, a master terminal implements the CSS-TS service endpoint (as defined in clause 9 of TS 103 286-2 [47]). A slave terminal or Companion Screen application connects to the CSS-TS service endpoint to establish a session of the Timeline Synchronization Protocol.

This protocol conveys messages containing setup-data and Control Timestamps and Actual, Earliest and Latest Presentation Timestamps that relate Wall Clock time to the Synchronization Timeline.

13.8.2 CSS-TS service endpoint (master terminal)

13.8.2.1 General

When the terminal is a master terminal, it shall implement an MSAS function that implements a CSS-TS service endpoint as defined in clause 9 of TS 103 286-2 [47]. The MSAS function of the master terminal shall provide the CSS-TS service endpoint on the master terminal's broadband interface.

For each media object involved in synchronization at the master terminal, the master terminal implements an SC function that interacts with the MSAS function of the master terminal to control the presentation timing and report the achievable presentation timings for that media object.

The MSAS function of the master terminal shall support a minimum of 10 simultaneous sessions of the Timeline Synchronization Protocol at the CSS-TS service endpoint and shall allow slave terminals and CSAs to connect to this service endpoint until the MSAS function of the master terminal has reached the limit of the number of simultaneous sessions it can support.

The MSAS function of the master terminal shall ignore the `Origin` header if it is present in the client handshake of the connection request.

When the terminal ceases to be a master terminal, it shall close any connections to the CSS-TS service endpoint from slave terminals or CSAs, following the process described in clause 9 of TS 103 286-2 [47].

The CSS-TS endpoint shall satisfy the security requirements of clause 11.7 of the present document.

13.8.2.2 Synchronization timeline availability

As the first stage of the protocol session, the MSAS function of the master terminal awaits a setup-data message from the slave terminal or CSA. This message requests the Synchronization Timeline to be used for the remainder of the protocol session. The Synchronization Timeline defines the reference frame for `contentTime` property values in Control Timestamps and Actual, Earliest and Latest Presentation Timestamps exchanged during the protocol session.

The requested Synchronization Timeline shall be available if the requirements for determining the availability defined in clause 9.7.3 of the present document and clause 9.2 of TS 103 286-2 [47] are met and the requested Timeline is supported by the master terminal (see clause 13.4.2) and the master terminal has sufficient resources to decode the requested Timeline (see clause 13.4.2).

NOTE 1: The availability of the Synchronization Timeline is dependent on whether the `contentIdStem` matches the `contentId` for the master content and whether the requested timeline is currently derivable for the master media.

NOTE 2: Availability can change during the session and this is reflected in the timestamp messages send by the MSAS function. For example: a change of `contentId` may change whether it matches the `contentIdStem` provided in the `setup-data` message.

NOTE 3: A timeline that is available when tuned to a DVB broadcast may not necessarily be available when playing back a PVR recording of that same broadcast unless the component carrying that timeline was recorded.

13.8.2.3 Frequency of control timestamp messages

The MSAS function of the master terminal shall send its first Control Timestamp within 500 ms of receiving the setup-data message and having determined the availability of the timeline.

NOTE: This means that the master terminal sends the first Control Timestamp at most 500ms after having played 2.5 seconds of the broadcast or SPTS stream subsequent to the request for a TEMI timeline being made, or within 500ms for all other types of timeline defined in the present document. For a TEMI timeline, if the content is not yet playing then this period will be longer.

EXAMPLE 1: A request is made to the MSAS function of the master terminal for a Period-Relative timeline for an MPEG DASH presentation. The request refers to a Period that is described in the MPD. The master terminal responds within 500ms of the request by sending a Control Timestamp giving the timeline position.

EXAMPLE 2: A request is made to the MSAS function of the master terminal for a TEMI timeline for a paused SPTS being streamed via broadband. The playback of the stream is not resumed until 10 seconds after the request is made. The master terminal therefore potentially waits up to 13 seconds (10 seconds plus 500ms plus 2.5 seconds) before responding with a Control Timestamp giving either the timeline position, or signalling the unavailability of the timeline.

The MSAS function of the master terminal shall send an updated Control Timestamp a minimum of 500 ms after a previous Control Timestamp when the synchronization timeline availability changes, or if the timeline is available and the timeline speed multiplier is non zero and the relationship between Wall Clock and Synchronization Timeline drifts by an amount exceeding ±1 ms when compared to the previously sent Control Timestamp. If there is a change of playback speed then the terminal shall send an additional Control Timestamp without waiting a minimum of 500 ms since the previous Control Timestamp.

The MSAS function of the master terminal shall be able to handle receiving a minimum of 2 Actual, Earliest and Latest Presentation timestamp messages per session per second.

13.8.2.4 Controlling timing of presentation

As described in clause 9 of TS 103 286-2 [47], the MSAS function of the master terminal exchanges timestamp messages with SC functions of CSAs and slave terminals to coordinate the synchronised presentation of content between the master terminal and the CSAs and slave terminals.

SC functions of slave terminals and CSAs send Actual, Earliest and Latest Presentation Timestamps that describe the range of timings of presentation that they can achieve. The MSAS function of the master terminal sends Control Timestamps to instruct the SC functions of slave terminals and CSAs as to the speed and timing of presentation that they need to adjust to in order to achieve presentation that is synchronised with the master media at the master terminal.

NOTE 1: Clauses C.5.3 and C.6.3 of TS 103 286-2 [47] provide guidance on how to calculate an achievable timing of presentation given Actual, Earliest and Latest Presentation Timestamps provided by SC functions of CSAs and slave terminals.

Control Timestamps sent by the MSAS function of the master terminal to slave terminals and CSAs shall, when the synchronization timeline is available, represent the timing of presentation of the master media by the master terminal. Control Timestamps shall represent timing of presentation with respect to the reference point for timestamping defined in clause 13.4.1 when the value of the `timelineSpeedMultiplier` property is 1 and may do so when the `timelineSpeedMultiplier` property has any other value.

NOTE 2: This relaxation of the requirement acknowledges that it is more challenging for a terminal to provide timing information that accurately represents the relationship between the timeline and the wall clock when playback is not at normal speed.

When a Control Timestamp is representing a timing of presentation with respect to the reference point for timestamping, it shall do so to within plus or minus the minimum synchronization accuracy defined in clause 9.7.4.

The `timelineSpeedMultiplier` property in Control Timestamps shall represent the speed of presentation of the master media at the master terminal. The property shall have the value 1 under normal playback conditions. When presentation of the master content is paused (by the user or by the terminal when waiting for buffers to fill), the value shall be zero. When presentation is moving at any other rate, the value shall be the approximate intended playback speed of the presentation (e.g. a value of 2 corresponds to x2 fast-forward). The true presentation speed may marginally differ from the intended playback speed reported in the property.

Adjustments to the timing of presentation of master media and any other media at the master terminal have the following requirements:

1) The MSAS function of the master terminal shall only adjust to a new timing of presentation if it is achievable for the master media and it is also achievable for other media being presented by the master terminal if it is performing multi-stream synchronization.

NOTE 3: As described in clause 13.5, a timing of presentation is achievable if the terminal can obtain the media sufficiently early or delay obtaining it or buffer it to present it sufficiently late.

2) The MSAS function of the master terminal should adjust the timing of presentation to be achievable by all slave terminals and CSAs that have supplied Earliest and Latest Presentation Timestamps, if such a timing is possible and it does not violate requirement 1).

NOTE 4: An achievable timing of presentation is any timing of presentation that falls within the interval described by Earliest and Latest Presentation Timestamps reported by slave terminals and CSAs.

3) If there is no timing of presentation that is achievable for all slave terminals and CSAs, then the MSAS function of the master terminal should adjust to a timing of presentation that is achievable by at least one slave terminal or CSA if such a timing is possible and it does not violate requirement 1).

If the existing timing of presentation of the master media is a timing that is already achievable by all slave terminals and CSAs then the MSAS function of the master terminal shall not adjust the timing of presentation.

NOTE 5: Slave terminals or CSAs can also supply an Actual Presentation Timestamp as part of Actual, Earliest and Latest Presentation Timestamp messages. The MSAS function can also take this into account in deciding on a timing of presentation. This will, in some situations, avoid a discontinuity in the presentation at the slave device and therefore provide a smoother viewing experience for the user.

NOTE 6: Slave terminals or CSAs can join or leave the synchronization at any time, by starting or stopping CSS-TS protocol sessions. Slave terminals or CSAs can also report new Actual, Earliest and Latest Presentation Timestamps. A master terminal is recommended to avoid unnecessarily adjusting the timing of presentation in response to these occurrences if possible. Every adjustment is a disruption that will be noticeable to the user.

The specific algorithm used to adjust the presentation timing and subsequently calculate the Control Timestamp sent to slave terminals and CSAs is implementation specific and outside the scope of the present document.

If the master terminal is also performing multi-stream synchronization and multi-stream synchronization fails and ceases, then the terminal shall also cease to be a master terminal.

13.8.3 SC function (slave terminal)

13.8.3.1 General

When the terminal becomes a slave terminal, the SC function of the slave terminal shall wait until the CSS-CII messages received from the master terminal have provided:

- a URL in the `tsUrl` property;
- an entry in the `timelines` property; and
- a `presentationStatus` property of "`okay`".

The terminal shall then commence the process of Timeline Synchronization in the role of a CSA via the CSS-TS interface (as defined in clause 9 of TS 103 286-2 [47]) if it has not already done so. When the terminal ceases to be a slave terminal, the SC function of the slave terminal may cease this process.

The SC function of the slave terminal shall connect to the service endpoint of the MSAS function that is located by the tsUrl property in CSS-CII messages received from the master terminal.

The SC function of the slave terminal shall establish this connection even if no media object has been added to the slave terminal `MediaSynchroniser`.

If any of the following situations occur, then the slave terminal shall deem that inter-device synchronization with the master terminal has failed and generate a permanent error with corresponding error code (defined in clause 8.2.3.2.4) for the `MediaSynchroniser` object and cease to be a slave terminal:

- The MSAS function refuses the request to connect to the CSS-TS service endpoint (error code 10).
- The CSS-TS service endpoint is, or becomes, unreachable or unresponsive (error code 6).
- The master terminal closes the connection (error code 10).
- The SC function does not receive a Control Timestamp within 5 seconds of sending the `setup-data` message to initialize the session (error code 10).

If none of the above situations occurs then the SC function of the slave terminal shall maintain the connection to the CSS-TS service endpoint until the terminal ceases to be a slave terminal (see clause 13.2.3) or multi-stream synchronization at the slave terminal fails (if the slave terminal is performing multi-stream synchronization).

13.8.3.2 Setup-data message

The SC function of the slave terminal shall use the following values for properties in the `setup-data` message that it sends at the start of the protocol session:

- The value of the `contentIdStem` property shall be the empty string.
- The value of the `timelineSelector` property shall be the value of the `timelineSelector` property obtained from the first JSON object listed in the timelines property of a CII message received from the master terminal. The CII message used shall be the most recent one that contained a timelines property.

NOTE 1: This message selects the timeline to be used as the Synchronization Timeline for the remainder of the protocol session. The value supplied by the master terminal in the CII message is the Timeline used for `contentTime` properties in Correlation Timestamps at the master terminal (see clause 13.6.2). The slave terminal is therefore setting the Synchronization Timeline to be the same as the Timeline used for the `MediaSynchroniser` API at the master terminal. It is also used as the Timeline for the `MediaSynchroniser` API at the slave terminal.

NOTE 2: The CII message also conveys the tick rate for the Synchronization Timeline in the same JSON object from which the `timelineSelector` property is obtained.

13.8.3.3 Sending Actual, Earliest and Latest Presentation Timestamps

If the SC function of the slave terminal has sent at least one Actual, Earliest and Latest Presentation Timestamp, then the SC function of the slave terminal shall send a new Actual, Earliest and Latest Presentation Timestamp if, and only if, all of the following are true:

- 500 ms has passed since the previous Actual, Earliest and Latest Presentation Timestamp was sent.
- The timing of presentation relative to the Wall Clock that is represented by any of the Actual, Earliest or Latest parts of the Timestamp message has changed by 1 ms or more.
- The most recent Control Timestamp received indicates that the presentation is playing at normal speed.

The SC function of the slave terminal shall not send an Actual, Earliest and Latest Presentation Timestamp message if the WC-Client function of the slave terminal calculates that the dispersion of the slave terminal Wall Clock is infinity (see clause 13.7.4).

The presentation timing represented by the Earliest and Latest parts of the Timestamp can change due to the addition or removal of a media object from the `MediaSynchroniser` or a change in the earliest and latest presentation timings possible for any one of the media objects currently added to the `MediaSynchroniser`. The presentation timing represented by the Actual Presentation Timestamp changes if the presentation timing of the media object it is derived from changes. How the Actual, Earliest and Latest Presentation Timestamp is calculated is defined in clause 13.8.3.4.

13.8.3.4 Value of Actual, Earliest and Latest Presentation Timestamps

An Actual, Earliest and Latest Presentation Timestamp message contains an Earliest and a Latest Presentation Timestamp and can contain an Actual Presentation Timestamp.

If an Actual Presentation Timestamp is included in the message then it shall consist of a point on the Synchronization Timeline (called a contentTime) and a time value of the Wall Clock. This timestamp shall represent the presentation

timing of one of the media objects being presented by the slave terminal with respect to the reference point for timestamping within plus or minus the minimum synchronization accuracy defined in clause 9.7.4.

NOTE 1: The contentTime is expressed in terms of the Synchronization Timeline and is therefore translated from being in terms of the timeline selected by the HbbTV® application for that media object by using the Correlation Timestamp most recently provided by the HbbTV® application for that media object.

If there are no media objects added to the slave terminal `MediaSynchroniser` then SC function of the slave terminal shall use negative infinity for the `wallClockTime` property of an Earliest Presentation Timestamp and positive infinity for the `wallClockTime` property of a Latest Presentation Timestamp.

An Earliest or a Latest Presentation Timestamp consists of a point on the Synchronization Timeline (called a contentTime) and a time value of the Wall Clock. The contentTime is also expressed in terms of the Synchronization Timeline.

A candidate Earliest Presentation Timestamp for a media object represents the earliest timing of presentation achievable for that media object taking into account the synchronization tolerance specified for it (see the `addMediaObject()` method) by subtracting the tolerance (converted to units of nanoseconds) from the Wall Clock time value part of the Timestamp.

A candidate Latest Presentation Timestamp for a media object represents the latest timing of presentation achievable for that media object also taking into account the synchronization tolerance specified for it (see the `addMediaObject()` method) by adding the tolerance (converted to units of nanoseconds) to the Wall Clock time value part of the Timestamp.

The SC function of the slave terminal shall use the Correlation Timestamp most recently provided by the HbbTV® application for a media object to calculate the `contentTime` property of candidate Earliest and Latest Presentation Timestamps in terms of the Synchronization Timeline.

NOTE 2: For a slave terminal MediaSynchroniser, the Timeline for the contentTime property in Correlation Timestamps is the same as the Synchronization Timeline and is the Timeline used for the contentTime property in Correlation Timestamps at the master terminal (see clause 13.4.3).

If there are media objects currently added to the slave terminal `MediaSynchroniser` then Earliest and Latest Presentation Timestamp messages that are sent to the MSAS function shall contain:

- the candidate Earliest Presentation Timestamp that represents a timing of presentation of media that is latest compared to the other candidate Earliest Presentation Timestamps; and
- the candidate Latest Presentation Timestamp that represents a timing of presentation of media that is earliest compared to the other candidate Latest Presentation Timestamps.

NOTE 3: Whether a slave terminal also includes an Actual Presentation Timestamp as part of the message is an implementation detail and is therefore outside the scope of the present document. As noted in clause 13.8.3.4, the inclusion of an Actual Presentation timestamp can assist in avoiding discontinuities in presentation at the slave terminal.

The set of candidate Earliest and Latest Presentation Timestamps that are to be used in the above process shall be the candidates corresponding to all media objects added to the slave terminal `MediaSynchroniser` if there exists some timing of presentation that is achievable for all these media objects. Otherwise the set of candidates used shall be those corresponding to any subset for which there exists some timing of presentation that is achievable for all media objects in the subset. The choice of the subset is implementation dependent.

NOTE 4: There is no achievable timing of presentation for all media objects if any of the candidate Earliest Presentation Timestamps represents a timing of presentation of media that is later than that represented by any of the candidate Latest Presentation Timestamps.

NOTE 5: The process of examining the range of achievable presentation timings for all participants is also performed by the MSAS function. Clauses C.5.3 and C.6.3 of TS 103 286-2 [47] provides examples of the calculations an MSAS function performs to do this.

13.8.3.5 Adjusting timing of presentation in response to Control Timestamps

Before the SC function of the slave terminal receives a first Control Timestamp message from the MSAS (in response to sending a setup data message), the SC function of the slave terminal has no knowledge of what time it should present content. At this stage, it shall therefore not change the playback state or adjust the presentation timing of any media objects that have been added to the slave terminal `MediaSynchroniser`.

The SC function of the slave terminal shall ignore Control Timestamps it receives and the requirements specified in the remainder of this clause shall not apply if there are no media objects added to the slave `MediaSynchroniser` when a Control Timestamp message is received, or if the dispersion calculated by the WC-Client function of the slave terminal is infinity.

A Control Timestamp consists of a point on the Synchronization Timeline (called a `contentTime`) and a time value of the Wall Clock. `contentTime` is expressed in terms of the Synchronization Timeline.

If the `contentTime` property of a Control Timestamp is a null value then the `MediaSynchroniser` of the slave terminal shall generate a permanent error with error code 15 for the `MediaSynchroniser` object (see clause 8.2.3.2.4) and cease to be a slave terminal.

For all media objects currently added to the slave terminal `MediaSynchroniser`, the SC function of the slave terminal shall adjust the presentation timing of the media object to match the timing expressed by the Control Timestamp to within plus or minus the greater of: the synchronization tolerance (specified for that media object) and the minimum synchronization accuracy specified in clause 9.7.4.

The timing of presentation of the media object as observed at the reference point for timestamping shall differ from the timing of presentation expressed in the Control Timestamp by no more than plus or minus the sum of:

- the value of the `interDeviceSyncDispersion` property of the `MediaSynchroniser` object (see clause 8.2.3.2.1) when it is next updated; and
- the greater of the synchronization tolerance and the minimum synchronization accuracy.

The SC function of the slave terminal shall determine the presentation timing for a media object by translating the contentTime property of the Control Timestamp to be in terms of the Timeline selected by the HbbTV® application for the media object using the Correlation Timestamp most recently provided by the HbbTV® application for that media object.

If it is not possible for presentation timing of a media object to be delayed or advanced sufficiently to achieve this then the `MediaSynchroniser` of the slave terminal shall generate a transient error with error code 1 or 11 as appropriate for the `MediaSynchroniser` object (as defined in clause 8.2.3.2.4).

If it becomes possible again for the timing of a media object to be delayed or advanced sufficiently (for example because a new Control Timestamp has been received, or streamed data for a broadband stream has become available sufficiently early) then the slave terminal shall adjust the timing of presentation of the media object to be synchronised and the `MediaSynchroniser` of the slave terminal shall generate an `onSyncNowAchievable` event (as defined in clause 8.2.3.2.3).

13.9 Trigger Events

The terminal is not required to implement the Trigger Events service endpoint as defined in clause 10 of TS 103 286-2 [47].

13.10 Sequence diagrams for timeline synchronization (Informative)

13.10.1 General

This clause provides a set of informational sequence diagrams to aid the understanding of the APIs for Timeline Synchronization between a master terminal and a Companion Screen application, or master terminal and a slave terminal acting in the role of a Companion Screen application. The architecture and master and slave roles are explained in clause 13.2. The API itself is specified in clause 8.2.3. Use of the API relate to the master and slave HbbTV® terminal's use of the following interfaces and their respective protocols specified in TS 103 286-2 [47]:

- Content Identification and other Information interface and protocol (CSS-CII).
- Wall Clock synchronization interface and protocol (CSS-WC).
- Timeline Synchronization interface and protocol (CSS-TS).

The service endpoints for these protocols are implemented by the master terminal. The HbbTV® applications running on both the master and slave terminals do not directly interact using these protocols, but instead direct the terminal to do so through use of the API.

Clauses 13.10.2, 13.10.3 and 13.10.4 illustrate, at a high level, the sequence of interactions between master and slave terminals and when the transition to and from being a master or slave terminal occurs.

Clauses 13.10.5, 130.10.6,13.10.7 and 13.10.8 illustrate, in greater detail, the inter-device synchronization protocol interactions between master and slave terminals. These clauses focus on the CSS-CII and CSS-TS protocols and their relationship to state changes of the various types of media objects (video/broadcast object, A/V Control object and HTML5 media element) when used as the master media at the master terminal and also on the use of multi-stream synchronization at the slave terminal. More messages may be sent via the protocols than those shown here. For example, these clauses do not detail the continuous exchange of Control Timestamps or Actual, Earliest and Latest Presentation Timestamps that maintain synchronization.

13.10.2 Initiation of timeline synchronization

Figure 24 shows a sequence diagram for the initiation of inter-device timeline synchronization between two terminals. Each terminal enters its respective role as a master and slave only from the point at which the HbbTV® applications instruct the MediaSynchroniser objects to enable the inter-device synchronization processes.

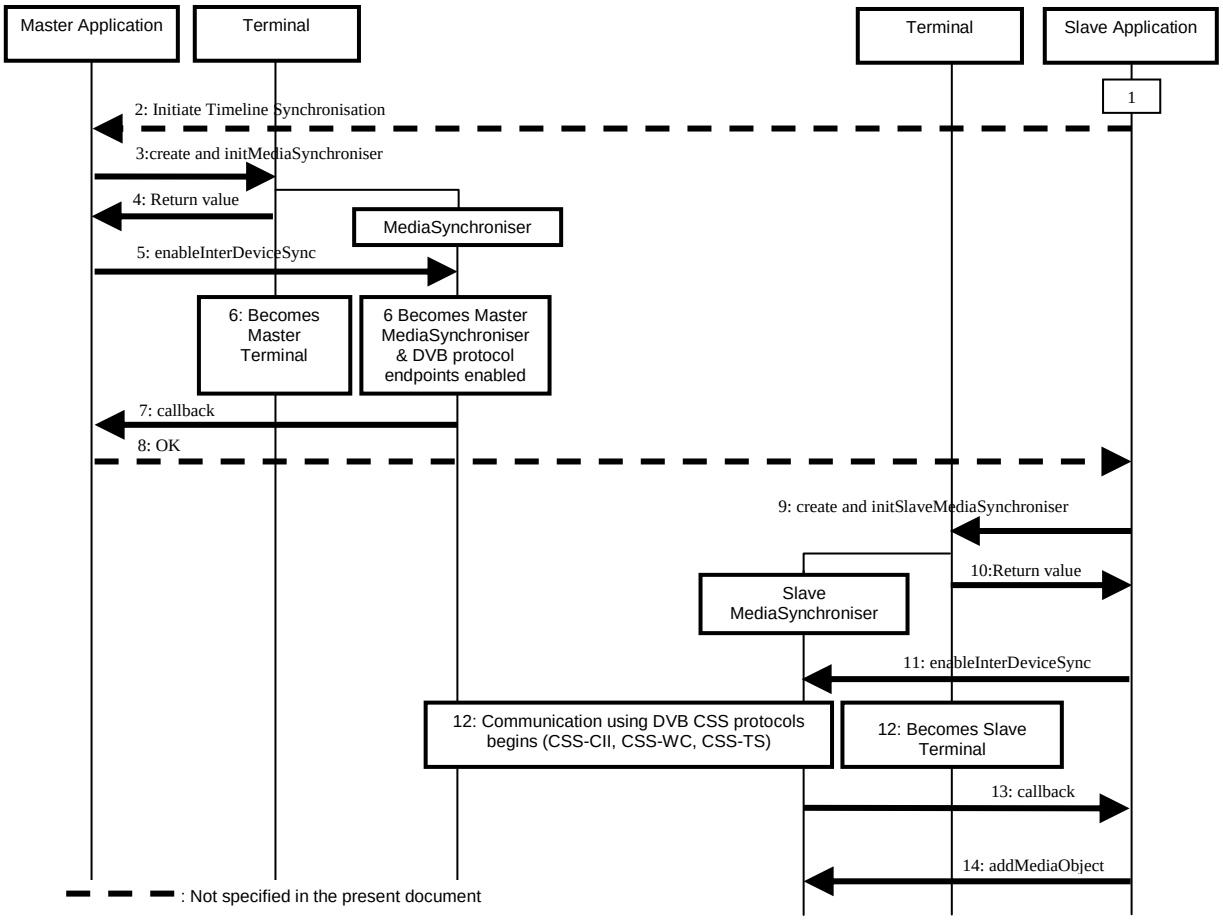


Figure 24: Initiation of inter-device timeline synchronization

1) The slave HbbTV® application is assumed to already be running and to have discovered the other terminal running the master HbbTV® application and the locations of the service endpoints it provides (such as CSS-CII and app to app communication).

2) Negotiation between master HbbTV® application and slave HbbTV® application to agree to perform inter-device synchronization is out of scope of this API and could therefore take place via proprietary app to app communication.

3) The master HbbTV® application creates a master `MediaSynchroniser` embedded object. The application initializes it using the `initMediaSynchroniser()` method, passing it an existing media object (video/broadcast, A/V Control or HTML5 media object) and a timeline specification. The terminal on which the master HbbTV® application is running is now a master terminal.

4) A `MediaSynchroniser` embedded object is returned to the master HbbTV® application.

5) The master HbbTV® application asks the `MediaSynchroniser` to enable inter-device synchronization functionality by calling the `enableInterDeviceSync()` method.

6) The terminal enables inter-device synchronization functionality by ensuring the protocol endpoints defined in TS 103 286-2 [47] (CSS-CII, CSS-WC and CSS-TS) are ready to accept connections. The terminal on which the master HbbTV® application is running is now a master terminal.

7) The master `MediaSynchroniser` confirms, by callback, to the master HbbTV® application that inter-device synchronization has been enabled.

8) The master HbbTV® application at this stage might choose to signal to the slave HbbTV® application that the master terminal is now ready to perform inter-device synchronization. How this is done is out of scope of this API and therefore could take place via proprietary app to app communication.

9) The slave HbbTV® application creates a `MediaSynchroniser` embedded object. The application initializes it using the `initSlaveMediaSynchroniser()` method, passing it the URL of the CSS-CII service endpoint on the master terminal.

10) A slave `MediaSynchroniser` embedded object is returned to the slave HbbTV® application. From this point the slave `MediaSynchroniser` begins to use the protocols to prepare for inter-device presentation (e.g. synchronizing its wall clock using the Wall Clock Synchronization Protocol).

11) The slave HbbTV® application asks the slave `MediaSynchroniser` to enable inter-device synchronization by calling the `enableInterDeviceSync()` method.

12) The terminal on which the slave HbbTV® application is running is now in the process of becoming a slave terminal. From this point the slave `MediaSynchroniser` begins to use the protocols defined in TS 103 286-2 [47] to prepare for inter-device presentation synchronization (e.g. communicating with the CSS-WC endpoint to estimate the master terminal Wall Clock and connecting to the CSS-TS endpoint to start exchanging timing information and synchronising its wall clock using the Wall Clock Synchronization Protocol). The terminal on which the slave HbbTV® application is running is now a slave terminal.

13) The slave terminal `MediaSynchroniser` confirms, by callback, to the slave HbbTV® application that inter-device synchronization has begun.

14) The slave HbbTV® application adds a media object to the slave MediaSynchroniser. The slave terminal now starts to use the timing information being exchanged via the protocols defined in TS 103 286-2 [47] to synchronise the presentation timing of the added media object to the presentation timing of media objects associated with the master `MediaSynchroniser`.

13.10.3 Protocols interactions for beginning inter-device synchronization

Figure 25 shows a sequence diagram for the protocol interactions for inter-device timeline synchronization between a master terminal and a slave terminal.

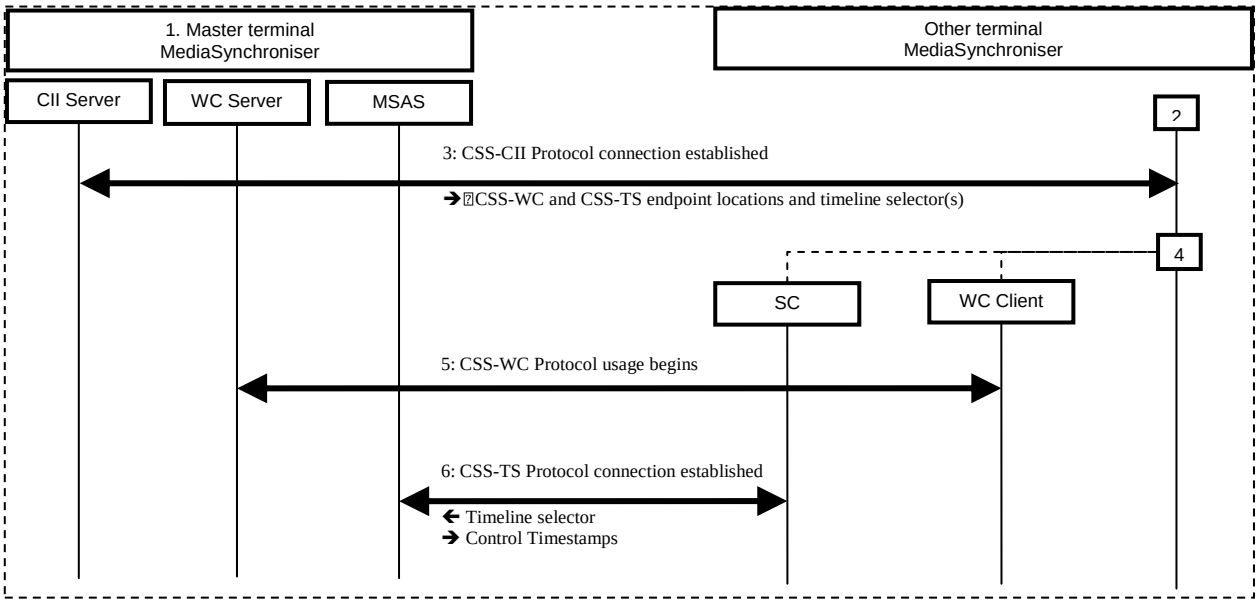


Figure 25: Protocol interactions for inter-device timeline synchronization

1) The application at the master terminal has already enabled inter-device synchronization and the servers implementing the endpoints for the CSS-CII, CSS-WC and CSS-TS protocols are already active.

2) The application at the other terminal enables inter-device synchronization.

3) The other terminal's `MediaSynchroniser` begins to interact with the endpoint for the master HbbTV® terminal that was provided to it by the slave HbbTV® application, using the CSS-CII protocol. CII messages sent by the master terminal via this protocol provide the locations of endpoints for the CSS-WC and CSS-TS protocols and the timeline selector for a timeline that can be requested from the master terminal.

4) The other terminal's `MediaSynchroniser` initializes any internal elementary functions it needs to perform Wall Clock synchronization using the CSS-WC protocol and Timeline Synchronization using the CSS-TS protocol.

5) The Wall Clock Client elementary function of the other terminal begins using the Wall Clock Synchronization Protocol to synchronise its Wall Clock to that of the master terminal.

6) The Synchronization Client elementary function of the other terminal connects to the CSS-TS endpoint and uses it to obtain Control Timestamps for the timeline identified by the timeline selector obtained earlier via the CSS-CII protocol.

The other terminal is a slave terminal once it has established the connections to CSS-CII and CSS-TS and begun to use the Wall Clock protocol. The slave terminal can now notify the application running on it, via callback, that inter-device synchronization is now enabled.

Once the slave terminal has approximated its Wall Clock to that of the master terminal, the slave HbbTV® terminal can then begin using the Timeline Synchronization Protocol to coordinate the timing of presentation of media objects associated with the slave terminal `MediaSynchroniser` with the timing of presentation of media objects associated with the master terminal `MediaSynchroniser`.

The use of these inter-device synchronization protocols continues indefinitely while inter-device synchronization is being performed in order to maintain synchronization.

TS 103 286-2 [47] describes the expected behaviour of a Synchronization Client elementary function of the slave terminal in terms of the CSS-TS protocol when it is trying to synchronise the presentation timing of zero, one or more media objects.

13.10.4 Termination of timeline synchronization

If a slave terminal disables inter-device synchronization, it stops using the protocols defined in TS 103 286-2 [47] and ceases to be a slave terminal. The master terminal continues unaffected.

However, if a master terminal disables inter-device synchronization, then its termination of protocol connections will cause slave terminals to also terminate inter-device synchronization and cease to be a slave terminal.

Figure 26 shows a sequence diagram for the termination of inter-device synchronization initiated by the master terminal.

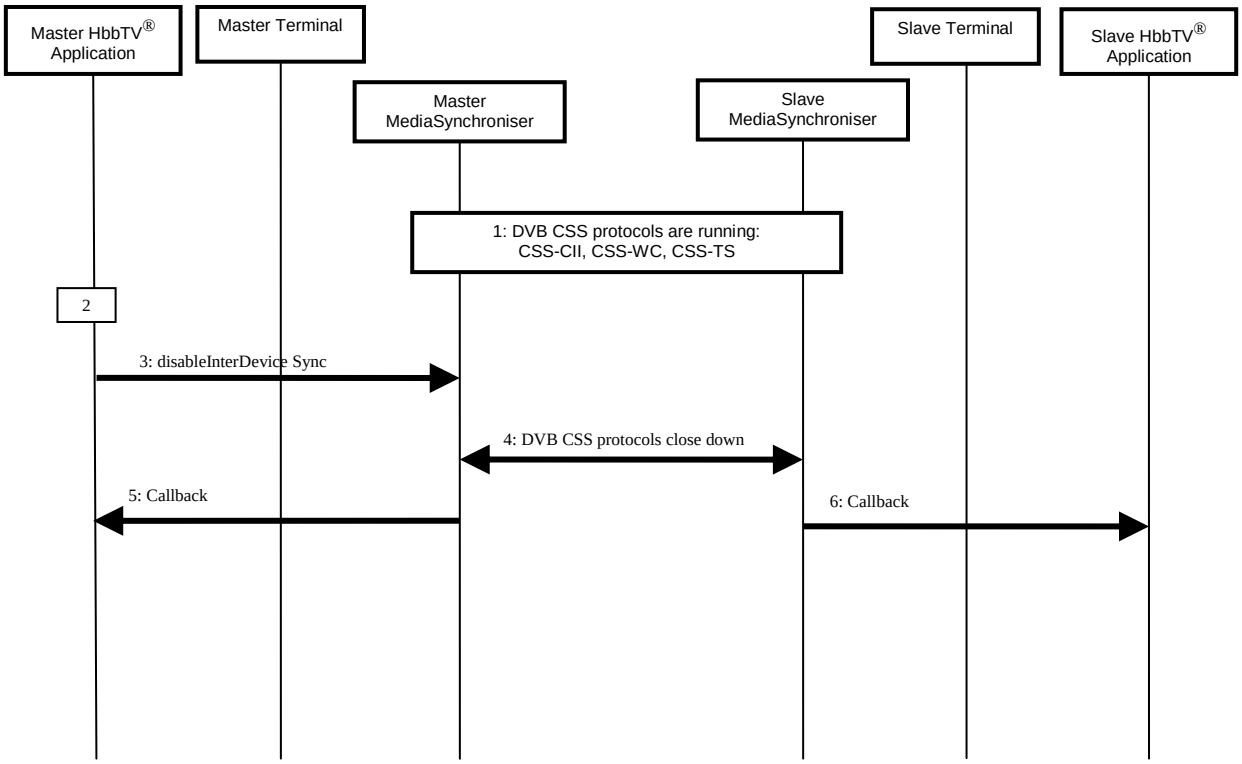


Figure 26: Termination of inter-device timeline synchronization

1) A slave terminal `MediaSynchroniser` object is currently communicating with a master terminal `MediaSynchroniser` object using the protocols to perform inter-device synchronization.

2) The master HbbTV® application decides to cease inter-device synchronization. How and when this decision is made is application specific behaviour.

3) The master HbbTV® application requests that the master terminal `MediaSynchroniser` disable inter-device synchronization.

4) The master terminal therefore closes down the protocol communication with any and all Companion Screen applications or slave terminal `MediaSynchronisers`.

5) The master terminal `MediaSynchroniser` informs the master HbbTV® application that inter-device synchronization has been disabled. The terminal is now no longer a master terminal.

6) The slave terminal `MediaSynchroniser` informs the slave HbbTV® application that inter-device synchronization was terminated by the master terminal. The terminal is now no longer a slave terminal.

13.10.5 Detailed protocol interaction (HTML5 media element presenting ISOBMFF as master media)

The following sequence of interactions is illustrated in figure Figure 27 with examples of message values in Table 24:

- Initially, the master terminal has already created an HTML5 media element to play an ISOBMFF media file streamed via HTTP (that is not a DASH presentation):
  - After the HTML5 element has buffered sufficient data and begins to play, the application initializes a `MediaSynchroniser` object, passing it that media object.
  - The master terminal begins to provide active endpoints for CSS-CII, CSS-WC and CSS-TS.
  - The master terminal then invokes the callback to notify the application that it is now a master terminal.

- Next, the application at the slave terminal initializes a slave `MediaSynchroniser` object and enables inter-device synchronization:
  - The slave terminal connects to the CSS-CII endpoint and the master terminal responds with a CII message (msg 1).
  - The slave terminal begins to use the CSS-WC protocol (not shown in the figure) by sending requests to the endpoint location given by the master terminal in msg 1.
  - The slave terminal also connects to the CSS-TS endpoint location given by the master terminal in msg 1.
  - At this stage the terminal can also notify the application, by callback, that it is now a slave terminal.
  - The slave terminal sends a setup-data message (msg 2) via the CSS-TS connection using the timeline selector obtained from msg 1.
  - The master terminal responds with a Control Timestamp message (msg 3) via the CSS-TS connection.
  - The slave terminal now has enough information to begin synchronising the presentation timing of any media objects it is presenting.
- Then at some later point, the application at the master terminal instructs the HTML5 media element to pause for a brief period:
  - The master terminal sends a Control Timestamp (msg 4) to notify the slave terminal.
  - When the application unpauses the HTML5 media element, normal playback speed is resumed. The master terminal sends an updated Control Timestamp (msg 5) to notify the slave terminal.
- When the HTML5 media element reaches the end of the media playback:
  - The master terminal sends a Control Timestamp (msg 6) to inform the slave terminal that playback has the appearance of having paused.
- Later, the application at the master terminal changes the `src` attribute of the HTML5 media element, or takes some other action that causes the media element to generate an error:
  - The `MediaSynchroniser` object at the master terminal enters into the permanent error state with error code 16.
  - The master terminal sends a CII message to the slave terminal informing it of the change in presentation status to "fault".
  - At the slave terminal, this results in a permanent error of the `MediaSynchroniser` with error code 6.
  - The master terminal then disconnects the CSS-CII and CSS-TS protocol connections and stops providing the endpoints.

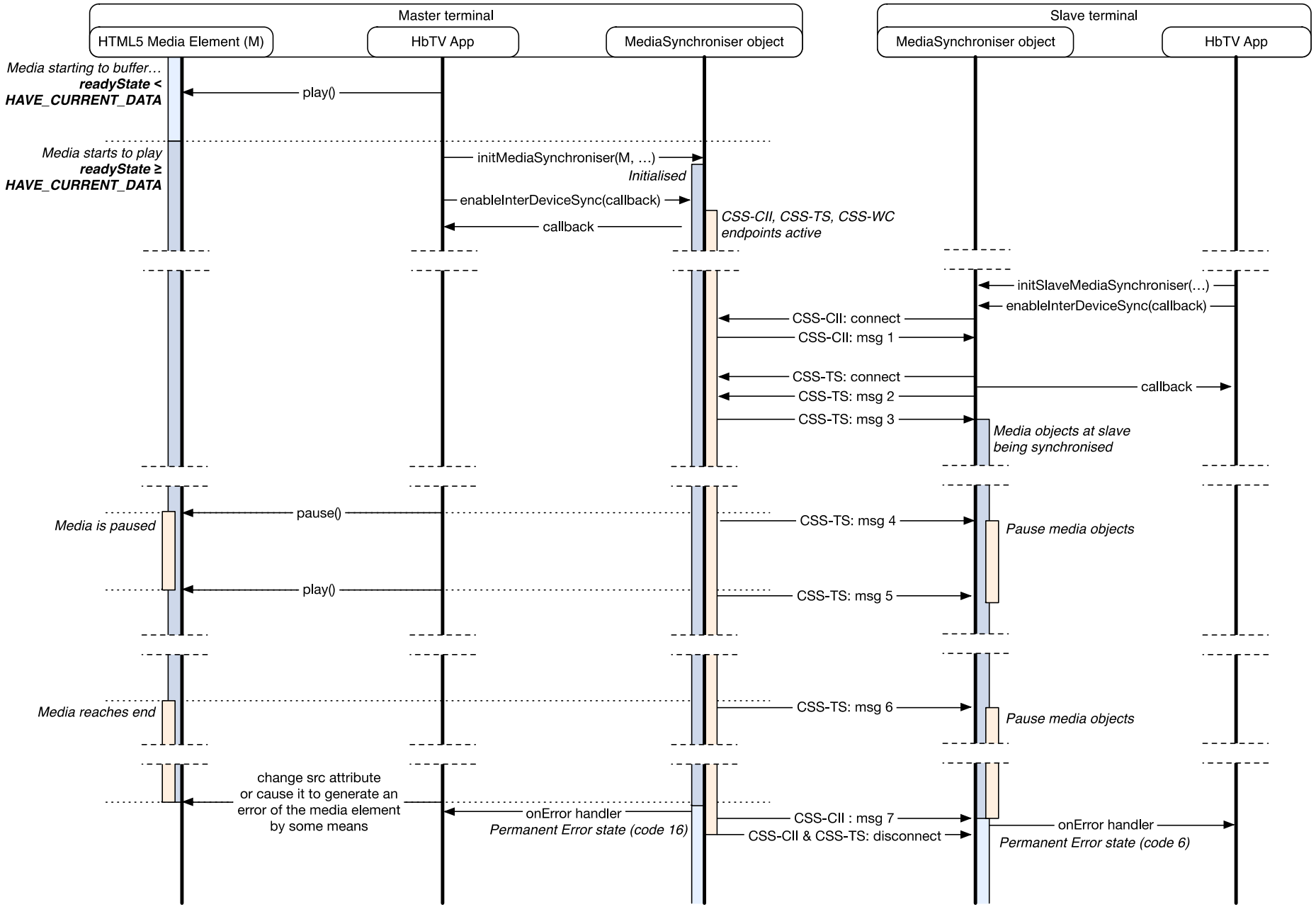


Figure 27: Inter-device synchronization sequence diagram where the master media is an HTML5 media element playing ISOBMFF media

Table 24: Inter-device synchronization messages where master media is

Message as referenced in figure Figure 27	Example JSON message contents

CSS-CII: msg 1	{
	"protocolVersion": "1.1",
	"contentId": "http://broadcaster.com/mystream.mp4",
	"contentIdStatus": "final",
	"presentationStatus": "okay",
	"mrsUrl" : null,
	"tsUrl" : "ws://192.168.1.5:7861/",
	"wcUrl" : "udp://192.168.1.5:6677",
	"teUrl" : null,
	"timelines" : [
	{
	"timelineSelector" : "urn:dvb:css:timeline:ct",
	"timelineProperties" : {
	"unitsPerTick": 1,
	"unitsPerSecond": 1000
	}
	}
	]
	}

CSS-TS: msg 2	{
	"contentIdStem" : "",
	"timelineSelector" : "urn:dvb:css:timeline:ct"
	}

CSS-TS: msg 3	{
	"contentTime" : "826",
	"wallClockTime" : "8576234985623",
	"timelineSpeedMultiplier" : 1
	}

CSS-TS: msg 4	{
	"contentTime" : "1523",
	"wallClockTime" : "8595262157800",
	"timelineSpeedMultiplier" : 0
	}

CSS-TS: msg 5	{
	"contentTime" : "1523",
	"wallClockTime" : "9485629346497",
	"timelineSpeedMultiplier" : 1
	}

CSS-TS: msg 6	{
	"contentTime" : "8192",
	"wallClockTime" : "8692746287477",
	"timelineSpeedMultiplier" : 0
	}

CSS-CII: msg 7	{
	"presentationStatus": "fault"
	}

an HTML5 media element playing ISOBMFF media

13.10.6 Detailed protocol interaction (A/V Control object presenting DASH as master media)

The following sequence of interactions is illustrated in figure Figure 28 with examples of message values in Table 25:

- Initially, the master terminal has already created an A/V Control object to play an MPEG DASH presentation:
  - The application calls the `play()` method of the A/V Control object causing it to enter the "connecting" state and, subsequently, the "buffering" state.
  - While the A/V Control object is undergoing these transitions, the application initializes a `MediaSynchroniser` object, passing it that media object and a timeline selector of "`urn:dvb:css:timeline:mpd:period:rel:25:00d1`".
  - The master terminal begins to provide active endpoints for CSS-CII, CSS-WC and CSS-TS.
  - The master terminal then invokes the callback to notify the application that it is now a master terminal.

- Next, the application at the slave terminal initializes a slave `MediaSynchroniser` object and enables inter-device synchronization:
  - The slave terminal connects to the CSS-CII endpoint and the master terminal responds with a CII message (msg 1). The presentation status conveyed in this message is "transitioning".
  - The slave terminal begins to use the CSS-WC protocol (not shown in the figure) by sending requests to the endpoint location given by the master terminal in msg 1.
  - The slave terminal also connects to the CSS-TS endpoint location given by the master terminal in msg 1.
  - At this stage the terminal can also notify the application, by callback, that it is now a slave terminal.
  - The slave terminal sends a setup-data message (msg 2) via the CSS-TS connection using the timeline selector obtained from msg 1.
  - The master terminal responds with a Control Timestamp message (msg 3) via the CSS-TS connection. This indicates that the timeline is currently at content time 0, but is currently paused (because the media is still buffering). The slave terminal now has enough information to begin synchronising the presentation timing of any media objects it is presenting (by seeking to the right time and pausing).
  - Shortly after, the A/V Control Object transitions to the "playing" state. This causes the master terminal to send an updated Control Timestamp (msg 4) and a CII message (msg 5) to inform the slave terminal that playback is now proceeding at normal speed and that the presentation status is now "okay".
- Then at some later point, the application at the master terminal instructs the A/V Control object to seek:
  - Because the required data is not yet buffered, this causes the A/V Control object to enter the "buffering" state.
  - The master terminal sends an updated Control Timestamp (msg 6) to notify the slave terminal.
  - When the A/V Control object has buffered sufficient media data and returns to the "playing" state, the master terminal sends an updated Control Timestamp (msg 7) to notify the slave terminal.
  - The seek operation has, in this instance, caused the A/V Control object to now be playing from a different period with id "00d2" of the DASH presentation. The master terminal sends a CII message (msg 8) to inform the slave terminal of the change to the content id.
- Later, the manifest for the DASH presentation is updated. This removes the period with id "00d1".
  - This results in a permanent error state of the `MediaSynchroniser` at the slave terminal with error code 15.
  - The timeline selector used by the slave terminal for the CSS-TS connection specified a timeline relative to the start of the period with id "00d1". The master terminal sends a Control Timestamp (msg 9) to inform the slave terminal that this timeline is no longer available.
  - This also results in a permanent error state of the `MediaSynchroniser` at the master terminal with error code 15.
  - The master and slave terminals disconnect the CSS-CII and CSS-TS protocol connections and the master terminal stops providing the endpoints.

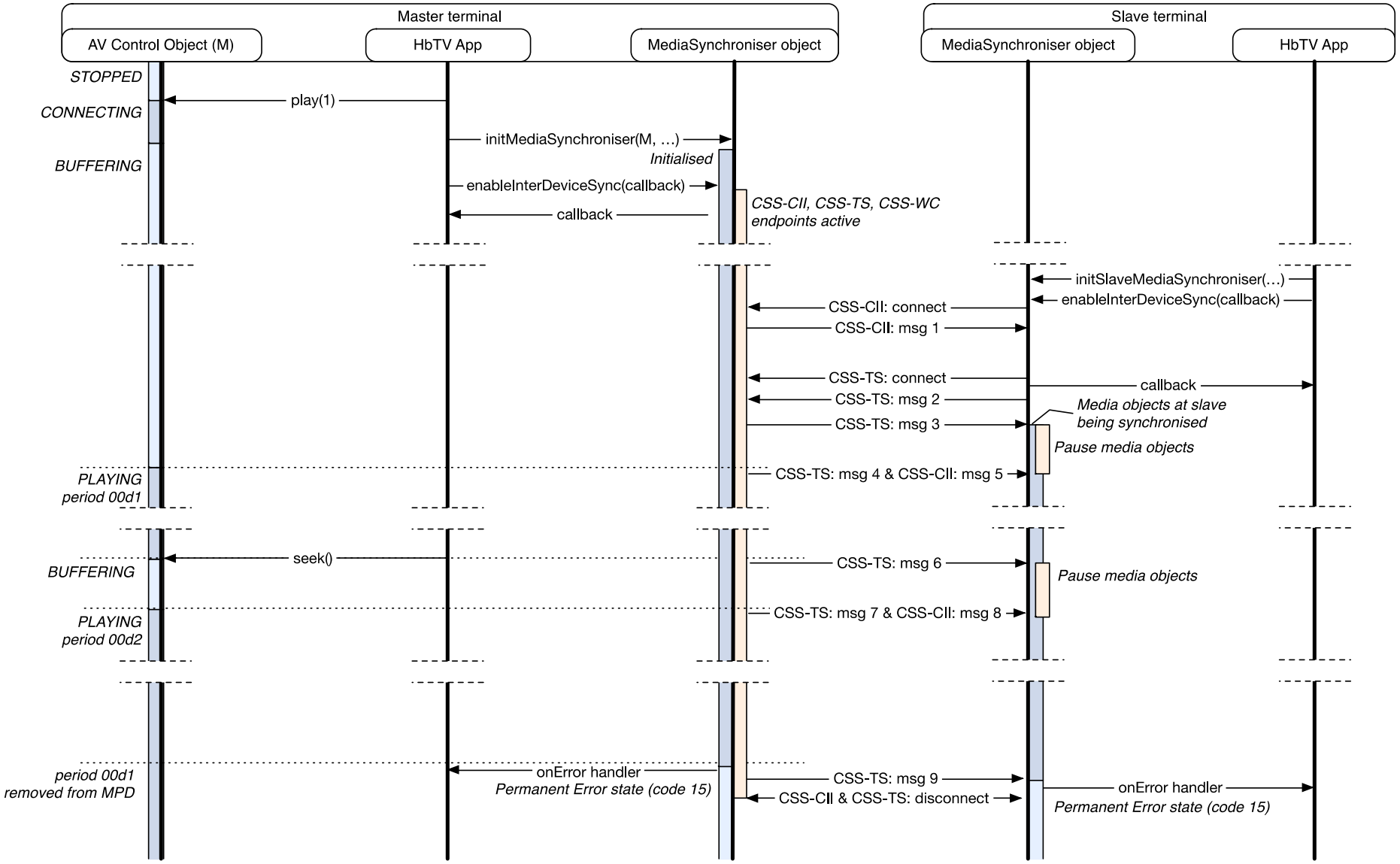


Figure 28: Inter-device synchronization sequence diagram where master media is an A/V Control object playing an MPEG DASH presentation

Table 25: Inter-device synchronization messages where master media is an A/V Control object playing an MPEG DASH presentation

Message as referenced in figure Figure 28	Example JSON message contents

CSS-CII: msg 1	{
	"protocolVersion": "1.1",
	"contentId": "http://broadcaster.com/mystream.mpd#period=00d1",
	"contentIdStatus": "final",
	"presentationStatus": "transitioning",
	"mrsUrl" : null,
	"tsUrl" : "ws://192.168.1.5:7861/",
	"wcUrl" : "udp://192.168.1.5:6677",
	"teUrl" : null,
	"timelines" : {
	"timelineSelector" : "urn:dvb:css:timeline:mpd:period:rel:25:00d1",
	"timelineProperties" : {
	"unitsPerTick": 1,
	"unitsPerSecond": 25
	}
	}
	}

CSS-TS: msg 2	{
	"contentIdStem" : "",
	"timelineSelector" : "urn:dvb:css:timeline:mpd:period:rel:25:00d1"
	}

CSS-TS: msg 3	{
	"contentTime" : "0",
	"wallClockTime" : "9000150284310",
	"timelineSpeedMultiplier" : 0
	}

CSS-TS: msg 4	{
	"contentTime" : "0",
	"wallClockTime" : "9008150224670",
	"timelineSpeedMultiplier" : 1
	}

CSS-CII: msg 5	{
	"presentationStatus": "okay"
	}

CSS-TS: msg 6	{
	"contentTime" : "175",
	"wallClockTime" : "9015150859370",
	"timelineSpeedMultiplier" : 0
	}

CSS-TS: msg 7	{
	"contentTime" : "15283",
	"wallClockTime" : "9016153926600",
	"timelineSpeedMultiplier" : 1
	}

CSS-CII: msg 8	{
	"contentId": "http://broadcaster.com/mystream.mpd#period=00d2",
	"contentIdStatus": "final"
	}

CSS-TS: msg 9	{
	"contentTime" : null,
	"wallClockTime" : "9017154281880",
	"timelineSpeedMultiplier" : null
	}

13.10.7 Detailed protocol interaction (video/broadcast object as master media)

The following sequence of interactions is illustrated in figure Figure 29 with examples of message values in Table 26:

- Initially, the master terminal has already created a video/broadcast object that is bound to the broadcast video being currently presented. The video/broadcast object is in the presenting state:
  - The application initializes a `MediaSynchroniser` object, passing it that media object and a timeline selector of "`urn:dvb:css:timeline:temi:1:1`" specifying a TEMI timeline from component tag 1 with timeline id 1.

- The master terminal begins to provide active endpoints for CSS-CII, CSS-WC and CSS-TS.
- The master terminal then invokes the callback to notify the application that it is now a master terminal.

- Next, the application at the slave terminal initializes a slave `MediaSynchroniser` object and enables inter-device synchronization:
  - The slave terminal connects to the CSS-CII endpoint and the master terminal responds with a CII message (msg 1). This presentation status conveyed in this message is "okay".
  - The slave terminal begins to use the CSS-WC protocol (not shown in the figure) by sending requests to the endpoint location given by the master terminal in msg 1.
  - The slave terminal also connects to the CSS-TS endpoint location given by the master terminal in msg 1.
  - At this stage the terminal can also notify the application, by callback, that it is now a slave terminal.
  - The slave terminal sends a setup-data message (msg 2) via the CSS-TS connection using the timeline selector obtained from msg 1.
  - The master terminal responds with a Control Timestamp message (msg 3) via the CSS-TS connection. The slave terminal now has enough information to begin synchronising the presentation timing of any media objects it is presenting.

- Then at some later point, there is a change in the DVB EIT present/following signalling in the broadcast for the DVB service currently being presented:
  - The master terminal sends a new content id in a CII message (msg 4) via the CSS-CII connection.

- Later, the application on the master terminal initiates a channel change using the `setChannel()` method of the video/broadcast object: (the application on the master terminal is assumed to remain running after the channel change completes):
  - The video/broadcast object transitions to the "connecting" state and the master terminal sends a CII message (msg 5) via the CSS-CII connection to indicate that the presentation status is "transitioning" and to provide a partial content id.
  - The change in presentation status to "transitioning" causes a transient error of the `MediaSynchroniser` at the slave terminal with error code 19.
  - Shortly after, the video/broadcast object transitions back to the "presenting" state as the channel change completes. The master terminal sends a CII message (msg 6) via the CSS-CII connection to indicate that the presentation status has reverted to "okay".
  - Within a second or two, the master terminal also determines that there is still a TEMI timeline available for the new service on the same component id and with the same timeline id as was specified in the timeline selector provided by the slave terminal in msg 2. The master terminal sends an updated Control Timestamp (msg 7) with the content time adjusted to match the new TEMI timeline.
  - The master terminal determines that it now has all information required to formulate a final version of the content id. The master terminal sends this final version to the slave terminal in a CII message (msg 8) via the CSS-CII connection.

- After some time, the terminal experiences a temporary signal loss.
  - Because this is a temporary signal loss only, the master terminal does not need to send any messages to the slave terminal.

- Finally, the application at the master terminal calls the release method of the video/broadcast object:
  - The video/broadcast object transitions to the "unrealized" state.
  - The `MediaSynchroniser` object at the master terminal enters into the permanent error state with error code 16.

- The master terminal sends a CII message (msg 9) to the slave terminal informing it of the change in presentation status to "fault".
- At the slave terminal, this results in a permanent error of the `MediaSynchroniser` with error code 6.
- The master terminal then disconnects the CSS-CII and CSS-TS protocol connections and stops providing the endpoints.

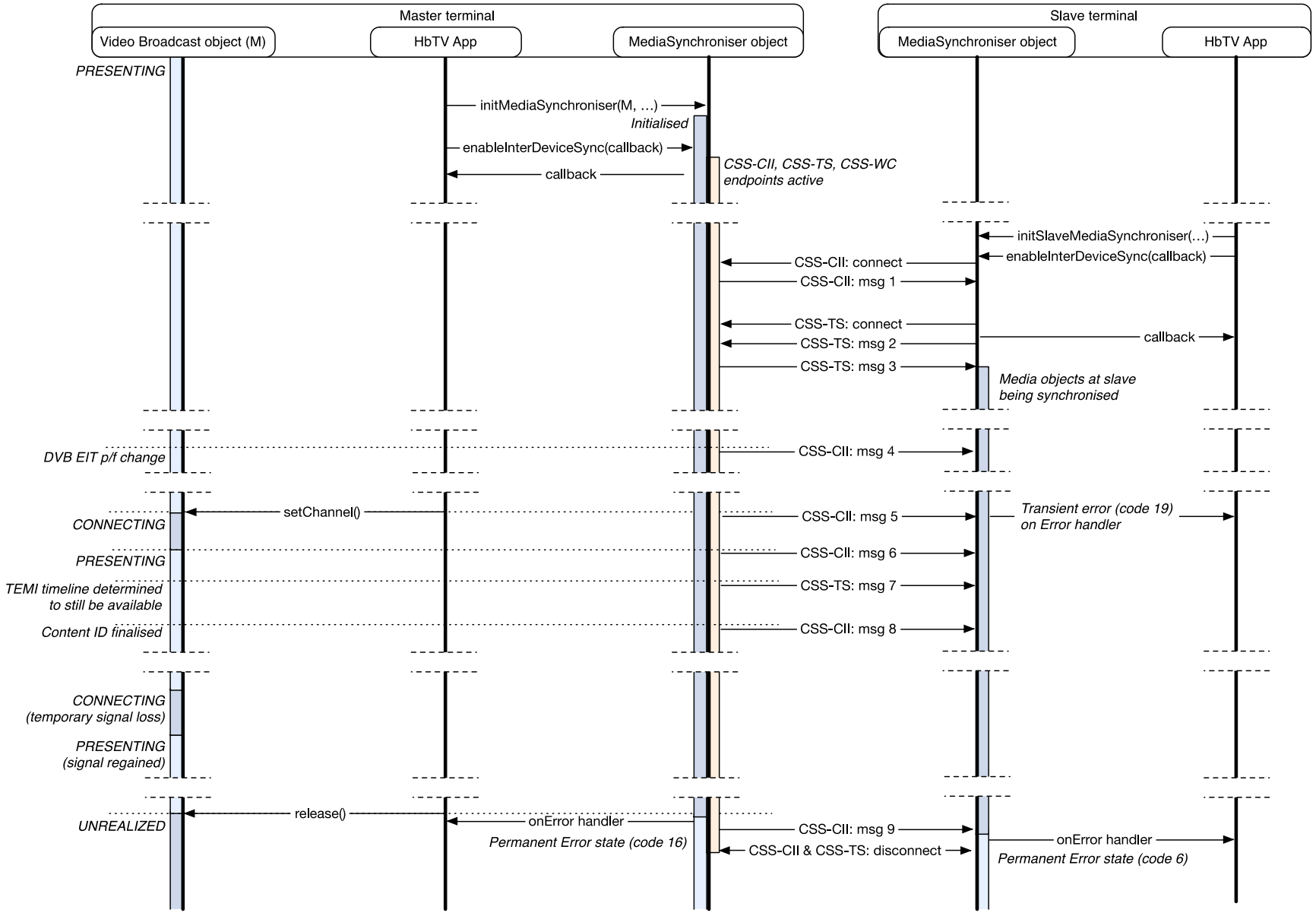


Figure 29: Inter-device synchronization sequence diagram where master media is a video/broadcast object

Table 26: Inter-device synchronization messages where master media is a video/broadcast object

Message as referenced in figure 29	Example JSON message contents

CSS-CII: msg 1	{
	"protocolVersion": "1.1",
	"contentId":
	"dvb://233a.1004.1044;35f7~20131004T0930Z--PT01H00M?nit_anc=495254",
	"contentIdStatus": "final",
	"presentationStatus": "okay",
	"mrsUrl" : null,
	"tsUrl" : "ws://192.168.1.5:7861/",
	"wcUrl" : "udp://192.168.1.5:6677",
	"teUrl" : null,
	"timelines" : {
	"timelineSelector" : "urn:dvb:css:timeline:temi:1:1",
	"timelineProperties" : {
	"unitsPerTick": 1,
	"unitsPerSecond": 50
	}
	}
	}

CSS-TS: msg 2	{
	"contentIdStem" : "",
	"timelineSelector" : "urn:dvb:css:timeline:temi:1:1"
	}

CSS-TS: msg 3	{
	"contentTime" : "18442500",
	"wallClockTime" : "9000150284310",
	"timelineSpeedMultiplier" : 1
	}

CSS-CII: msg 4	{
	"contentId":
	"dvb://233a.1004.1044;35f8~20131004T1030Z--PT00H30M?nit_anc=495254",
	"contentIdStatus": "final"
	}

CSS-CII: msg 5	{
	"contentId": "dvb://233a.1004.1080",
	"contentIdStatus": "partial",
	"presentationStatus": "transitioning"
	}

CSS-CII: msg 6	{
	"presentationStatus" : "okay"
	}

CSS-TS: msg 7	{
	"contentTime" : "2900015",
	"wallClockTime" : "9000183280003",
	"timelineSpeedMultiplier": 1
	}

CSS-CII: msg 8	{
	"contentId": "dvb://233a.1004.1080;21af~20131004T1015Z--PT01H00M",
	"contentIdStatus": "final"
	}

CSS-CII: msg 9	{
	"presentationStatus": "fault"
	}

13.10.8 Detailed protocol interaction (two media objects at the slave terminal)

The following sequence of interactions is illustrated in figure Figure 30 with examples of message values in Table 27:

- Initially, the master terminal has already enabled inter-device synchronization and the application at the other terminal has created a video/broadcast object that is bound to the broadcast video being currently presented and also an A/V Control Object for audio delivered via MPEG DASH but is currently in a paused state:
  - The application at the other terminal initializes a `MediaSynchroniser` object, passing it the URL for the CSS-CII service endpoint provided by the master terminal.
  - The application also uses the `addMediaObject()` method to add the video/broadcast object to the `MediaSynchroniser`.

- At this stage, there is no synchronization taking place.

- Next, the application enables inter-device synchronization using the `initSlaveMediaSynchroniser()` method:
  - The terminal connects to the CSS-CII endpoint and the master terminal responds with a CII message (msg 1).
  - The terminal begins to use the CSS-WC protocol (not shown in the figure) by sending requests to the endpoint location given by the master terminal in msg 1.
  - The terminal also connects to the CSS-TS endpoint location given by the master terminal in msg 1.
  - At this stage the terminal can also notify the application, by callback, that it is now a slave terminal and is performing inter-device synchronization.
  - The slave terminal sends a setup-data message (msg 2) via the CSS-TS connection using the timeline selector obtained from msg 1.
  - The master terminal responds with a Control Timestamp message (msg 3) via the CSS-TS connection. The slave terminal now has enough information to begin synchronising the presentation timing of the video/broadcast object.

- At a later point, the application at the slave terminal adds the A/V Control object to the `MediaSynchroniser`:
  - The slave terminal observes that given the most recent Control Timestamp received from the master terminal, it cannot obtain the media stream data early enough to achieve the required presentation timing for the MPEG DASH presentation. This causes a transient error of the `MediaSynchroniser` with code 1, listing the A/V Control object as the media object with the error condition.
  - The slave terminal chooses to send an Actual, Earliest and Latest Presentation timestamp message (msg 4) via the CSS-TS connection to the master terminal. The timestamps within this message reflect the limited range of presentation timings that are achievable for both the video/broadcast object and the A/V Control object. In this situation the slave terminal has a buffer for media synchronization to allow it to buffer broadcast video.
  - The master terminal chooses to adjust its timing of presentation in response to receiving msg 4. The master terminal then sends a new Control Timestamp message (msg 5) via the CSS-TS connection to the slave terminal reflecting this new presentation timing.
  - The slave terminal examines the new Control Timestamp and determines that it can now achieve synchronization for the A/V Control object and therefore an `onSyncNowAchievable` event of the `MediaSynchroniser` is generated. The A/V Control object is transitioned briefly through the "buffering" state and onto the "playing" state.
  - Because the slave terminal has a buffer for media synchronization, it is able to adjust the presentation timing of the video/broadcast object to keep it synchronised.

- Finally, the application at the slave terminal disables inter-device synchronization:
  - This causes the `MediaSynchroniser` at the slave terminal to cease synchronising the video/broadcast object and A/V Control objects.
  - The callback is then used to notify the application that the terminal has ceased being a slave terminal.
  - The application chooses to stop the A/V Control object playing but to leave the video/broadcast object playing.

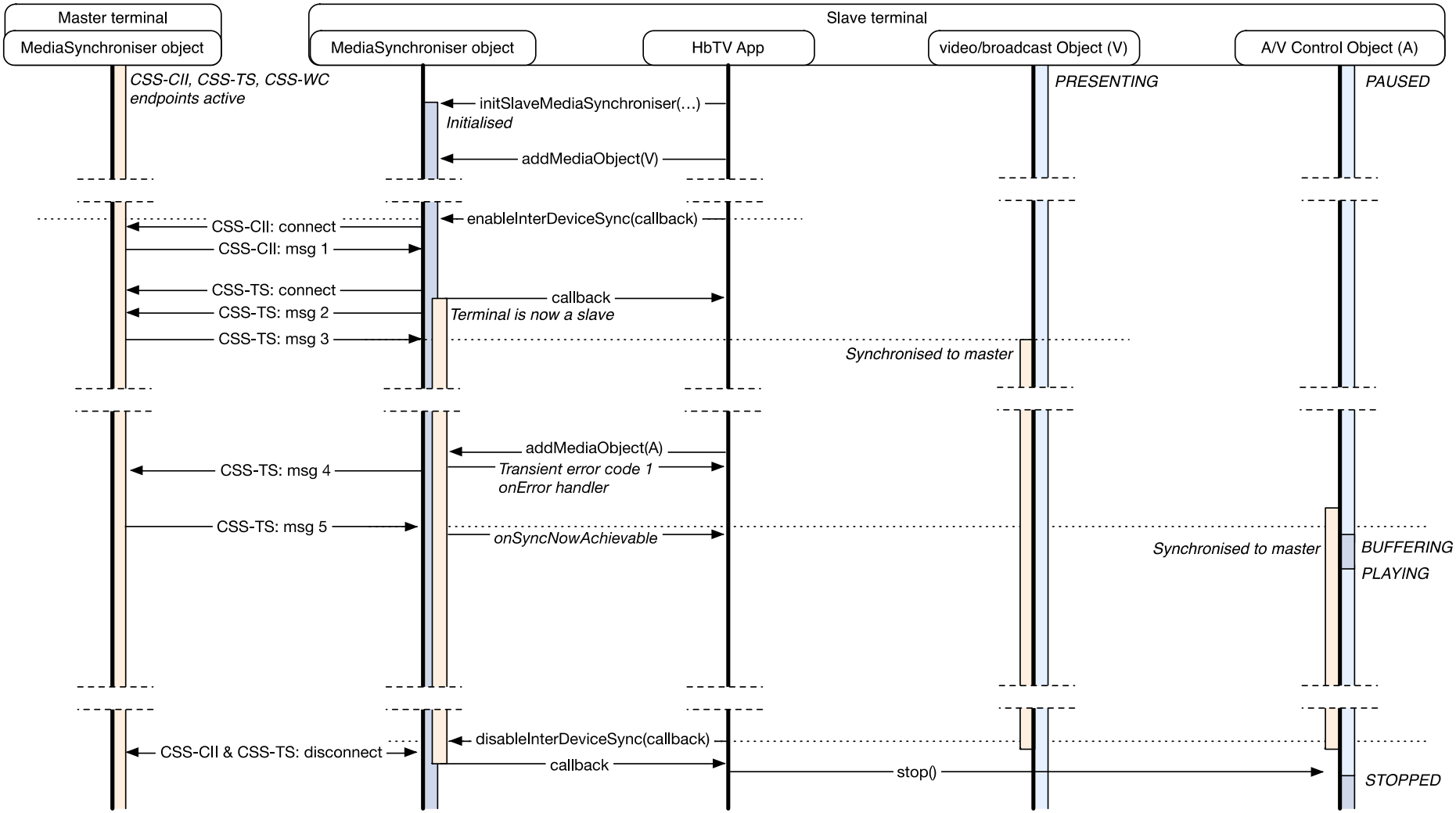


Figure 30: Inter-device synchronization sequence diagram where there are two media object synchronised at a slave terminal

Table 27: Inter-device synchronization messages where there are two media object synchronised at a slave terminal

Message as referenced in figure 30	Example JSON message contents

CSS-CII: msg 1	{
	"protocolVersion": "1.1",
	"contentId":
	"http://mybroadcaster.com/mystream.mp4",
	"contentIdStatus": "final",
	"presentationStatus": "okay",
	"mrsUrl" : null,
	"tsUrl" : "ws://192.168.1.5:7861/",
	"wcUrl" : "udp://192.168.1.5:6677",
	"teUrl" : null,
	"timelines" : {
	"timelineSelector" : "urn:dvb:css:timeline:ct",
	"timelineProperties" : {
	"unitsPerTick": 1,
	"unitsPerSecond": 25
	}
	}
	}

CSS-TS: msg 2	{
	"contentIdStem" : "",
	"timelineSelector" : "urn:dvb:css:timeline:ct"
	}

CSS-TS: msg 3	{
	"contentTime" : "100580",
	"wallClockTime" : "9000150284310",
	"timelineSpeedMultiplier" : 1
	}

CSS-TS: msg 4	{
	"earliest": {
	"contentTime" : "100674",
	"wallClockTime" : "9000150284310"
	},
	"latest": {
	"contentTime" : "100674",
	"wallClockTime" : "9008150284310"
	}
	}

CSS-TS: msg 5	{
	"contentTime" : "100680",
	"wallClockTime" : "9000150284310",
	"timelineSpeedMultiplier" : 1
	}

13.11 Application to media synchronization

13.11.1 General

The terminal shall support the following ways for an application to obtain the current media playback position for media objects (HTML5 media elements, video/broadcast objects and the A/V Control object):

1) The properties of the media objects that expose current media playback position as described in clause 13.11.2.

2) The `currentTime` property of a `MediaSynchroniser` object that exposes the current media timeline position as described in clause 13.11.3.

NOTE: Using the `MediaSynchroniser` object enables an application to select the timeline used to report the current playback position relative to a timeline even if no media synchronization is done.

13.11.2 Reading the media playback position of media objects

Each object or element that enables an application to present media provides a property that enables an application to read the current media playback position. These are as follows:

- For the HTML5 media elements, the `currentTime` property.
- For the A/V Control object , the `playPosition` property.

NOTE 1: The video/broadcast object does not provide a playback position, but applications can use the `MediaSynchroniser` object to retrieve the position on a timeline used for media synchronization as defined in clause 13.11.3.

When an application reads one of these properties, the value returned shall be the time of the last video frame that was composed with graphics before the method was called and shall be accurate to within 100ms. For the A/V Control object, the value returned shall be updated each time the property is read. The precision of the playback position shall at least correlate with either:

- the frame rate of the video component presented by the media object, i.e. it is at least 40 ms for 25 fps video and 20 ms for 50 fps; or
- the length of an access unit of the audio component presented by the media object, e.g. is at least 24 ms for MPEG 1 Layer 2 at 48 kHz sample rate or 42,67 ms for HE-AAC at 48 kHz sample rate.

For the A/V Control object:

- For on-demand content the value returned when the property is read shall be as defined in clause 8.2.5.1 of the OIPF DAE specification [1].
- For MPEG-DASH the value returned when the property is read shall be as defined in clause 9.4.3 of the present document.

For HTML5 media elements, the value returned when the property is repeatedly read is defined by that specification - see the description of the 'official playback position' concept.

NOTE 2: The properties encode returned values in different ways - seconds for the `currentTime` property (milliseconds can be expressed in the fraction part of the returned value) and integer milliseconds for the `playPosition` property. This has no effect on the value that is returned.

13.11.3 Reading the media playback position of the MediaSynchroniser object

Using the `MediaSynchroniser` API (defined in clause 8.2.3), an application can create a `MediaSynchroniser` object using a given media object as the master media (see clause 13.2.4). In doing so it can specify a timeline to use that is derived from that media object's media stream. The `currentTime` property of the `MediaSynchroniser` object reports current playback position in terms of that timeline.

When an application reads the `currentTime` property of a `MediaSynchroniser` object (see clause 8.2.3.2.1) the returned value shall correspond to the current position of the timeline used by the `MediaSynchroniser` object:

- For a `MediaSynchroniser` object that has been initialized by calling the `initMediaSynchroniser()` method, the returned value shall correspond to the current playback position of the media object that was passed as an argument to the `initMediaSynchroniser()` method (the master media). The value returned shall be the time of the last video frame that was composed with graphics before the property was queried and shall be accurate to within 100 ms. The precision of the playback position shall be at least correlate with either:
  - the highest frame rate of any video being presented on the terminal where that video is a component of a media object attached to the `MediaSynchroniser`, e.g. it is at least 40 ms for 25 fps video and 20 ms for 50 fps; or
  - if there is no applicable video component, the shortest length of an access unit of any audio being presented on the slave terminal where that audio is a component of a media object attached to the `MediaSynchroniser`, e.g. is at least 24 ms for MPEG 1 Layer 2 at 48 kHz sample rate or 42,67 ms for HE-AAC at 48 kHz sample rate.

- For a `MediaSynchroniser` object that has been initialized by calling the `initSlaveMediaSynchroniser()` method, the value returned shall correspond to the current playback position of the media object on the master terminal that was passed as an argument to the `initMediaSynchroniser()` method on the master terminal (the master media). The precision of the playback position shall be at least correlate with either:

  - the highest frame rate of any video being presented on the slave terminal where that video is a component of a media object attached to the `MediaSynchroniser`, e.g. it is at least 40 ms for 25 fps video and 20 ms for 50 fps; or

  - if there is no applicable video component, the shortest length of an access unit of any audio being presented on the slave terminal where that audio is a component of a media object attached to the `MediaSynchroniser`, e.g. is at least 24 ms for MPEG 1 Layer 2 at 48 kHz sample rate or 42,67 ms for HE-AAC at 48 kHz sample rate; or

  - if there are no applicable video or audio components then it shall be at least 100 ms.

14 Companion screens

14.1 Introduction

This clause introduces the methods to allow for interaction between HbbTV® and Companion Screens.

Whilst primarily targeted at iOS™ and Android™ devices, the framework described here should allow Companion Screens of any type to be used.

The HbbTV® terminal and the Companion Screens have to be connected to the same local network, and the local network should be connected to the Internet.

14.2 Description of framework (informative)

14.2.1 Supported features

This clause is written to allow for the following features:

- An HbbTV® application launching a Companion Screen application.

  The Companion Screen application may be an HTML application running in a browser on the Companion Screen, or may be a native Companion Screen application. There is also the facility for the HbbTV® application to direct the user to the location of a native application in a Companion Screen's 'store' (so that application can be downloaded) if it is not already installed on the user's Companion Screen device.

- A Companion Screen application launching a broadcast independent HbbTV® application on an HbbTV® terminal.

- To allow an HbbTV® application and a Companion Screen application to communicate directly by establishing a communication channel onto which text or binary messages can be exchanged, regardless of the launch methods of either the HbbTV® application or the Companion Screen application.

- To enable a companion screen or another HbbTV® terminal to locate the services and User Agent String, provided by the HbbTV® terminal, that can then be used via the methods described in clause 14.

14.2.2 Model

14.2.2.1 Launching a companion screen application

Figure 31 provides an architecture for launching a CS application.

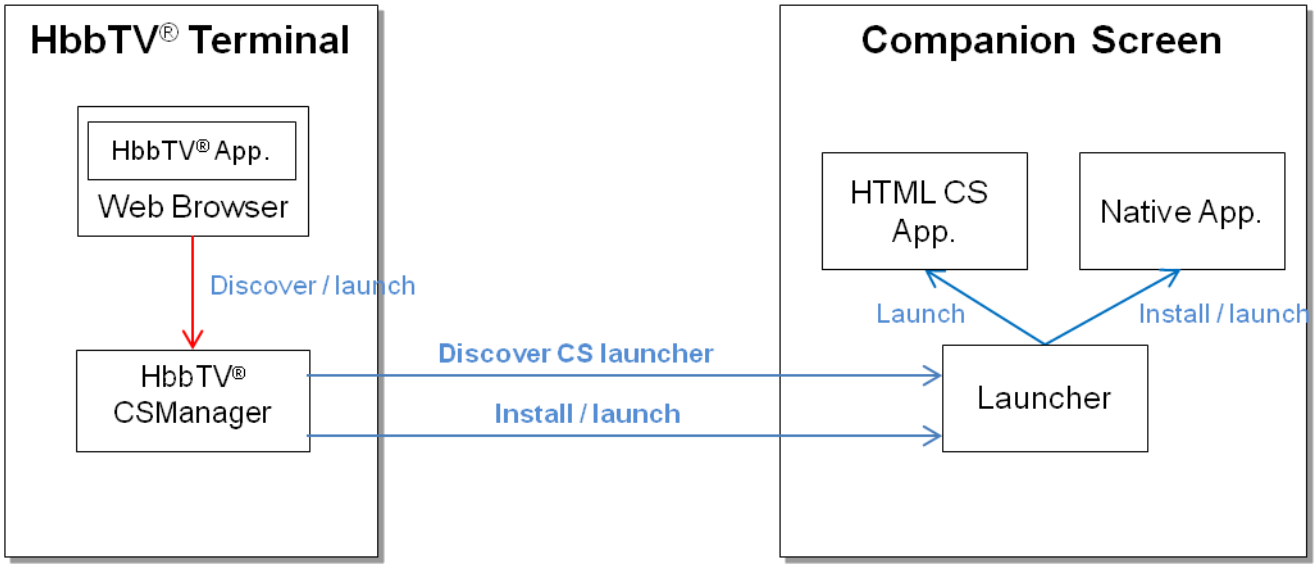


Figure 31: Architecture for launching a CS application

The following functions are distinguished in this architecture:

- Web Browser: running the web application consists of HTML5 and associated web technologies. The application environment for the HbbTV® terminal is described in clause 8.
- HbbTVCSManager: resides in the HbbTV® terminal. HbbTVCSManager is responsible for discovering the Companion Screens with a running Launcher and sending application launch/install information to the Launcher. The HbbTVCSManager embedded object is defined in clause 8.2.6.1.
- Native application: runs on specific Companion Screen platforms (e.g Android™, iOS™, etc.). Usually the native application is running as a binary code.
- Launcher: resides in the Companion Screens. The Launcher is responsible for communicating with HbbTVCSManager and launching and/ or installing the CS application. The communication protocol is not defined in the present document. The requirements on the Launcher are described in clause 14.3.

For an HbbTV® application to launch a CS application, the components defined above are used in the following way:

- The HbbTV® application, using the API defined here, discovers Companion Screens with a running Launcher application that are associated with the terminal on which the HbbTV® application is running.
- The HbbTV® application obtains platform information, including the OS that the Companion Screen is running. The HbbTV® application is then in a position to (in an application specific way) determine the launch (or installation) URL of the CS application it wants to launch (or install) on each Companion Screen.
- The HbbTV® application can then, using another API defined here, send each Launcher application the launch / install information specific to that Companion Screen.

This can be repeated for each Companion Screen that the HbbTV® application wants to utilise.

The Launcher application will then attempt to launch or install the CS application given the information provided to it by the HbbTV® application. However it may prevent the CS application from launching / installing for a variety of reasons, for example:

- The Launcher application has automatically blocked the launch (perhaps from a historical user decision to always block this application).
- The Launcher application has asked the user for permission to launch the application this time, but it was explicitly denied by the user.

- The Launcher application has asked the user for permission to launch the application this time, but did not get a response either way from the user, and has timed out.

14.2.2.2 Application to application communication

Figure 32 provides an architecture for application to application communication.

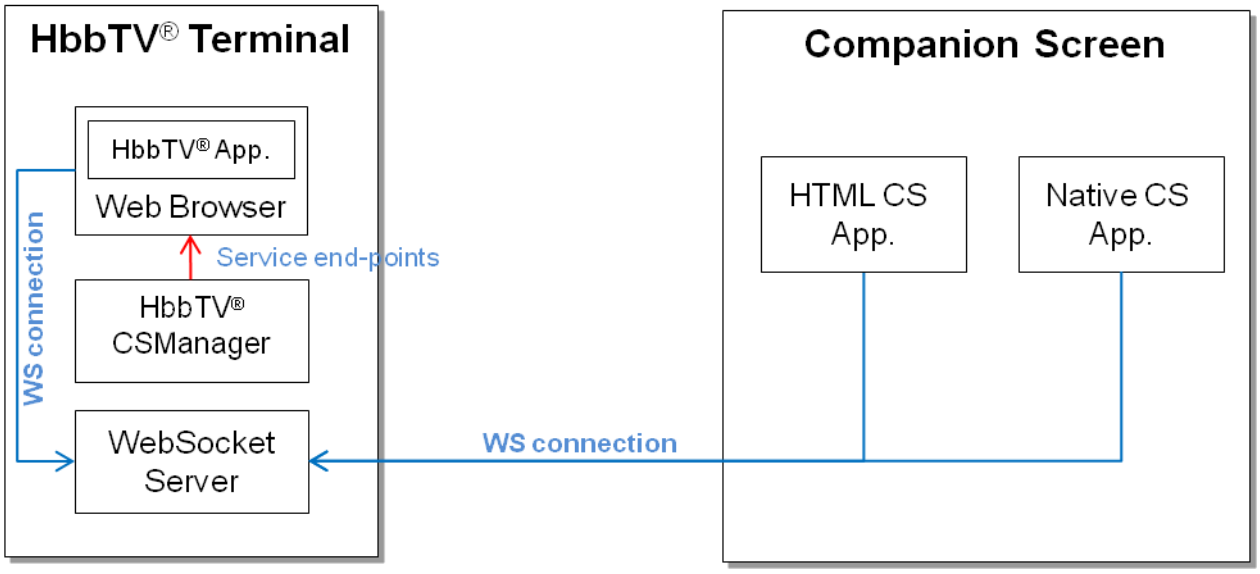


Figure 32: Architecture for application to application communication

The following functions are distinguished in this architecture.

- HbbTVCSManager: HbbTVCSManager is responsible for providing service endpoints for application to application communications. The API is defined in clause 8.2.6.1.

- Web Socket Server: resides in the HbbTV® terminal. Web Socket Server is responsible for handling web socket connections both from the HbbTV® application and the CS application. The communications between the applications is described in clause 14.5.

For an HbbTV® application to directly communicate with a CS application, there are two aspects described in the present document. The first is describing the methods used to discover the service endpoints. The second is to describe how to attach and communicate over the service once this has been discovered.

For discovering the application to application communication service endpoint, there are three methods described:

- HbbTV® applications may use an API (defined within the present document) to discover the location of the service endpoint.

- CS applications may use one of the following two methods:

  - If the CS application has been launched by an HbbTV® application (as described in clauses 14.3 and 14.4), then the location of the service endpoint may have been provided to the CS application as one of its launch parameters.

  - If the CS application has been launched independently, then it has to discover the location of the service endpoint using the mechanisms described in clause 14.7.

The application to application communication service is provided by a Web Socket Server located on the terminal and so for attaching and communicating over the service a web socket client API may be used.

The Web Socket Server receives requests for connections from the HbbTV® application and from a Companion Screen or another HbbTV® terminal. Pairing rules are defined to enable the server to establish a link between one connection from the HbbTV® application and one connection from the Companion Screen or other HbbTV® terminal. The server then acts as a relay passing the information from one connection to the other.

14.2.2.3 Remotely launching HbbTV® applications

Figure 33 provides an architecture for remotely launching HbbTV® applications.

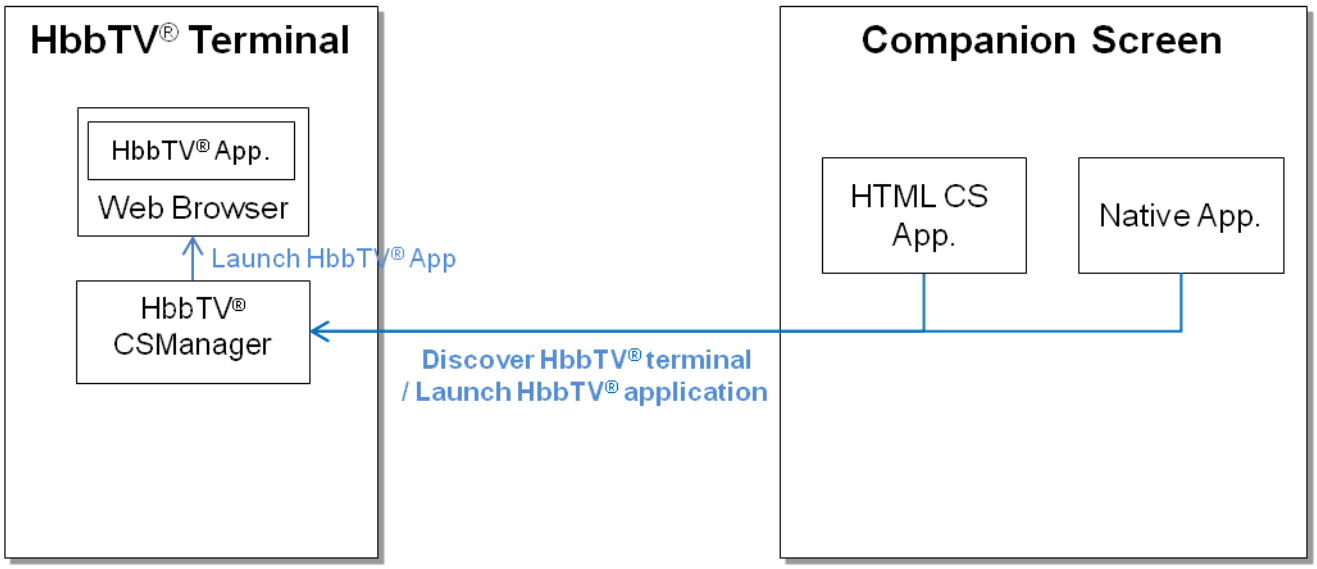


Figure 33: Architecture for remotely launching HbbTV® applications

The following functions are distinguished in this architecture.

- HbbTVCSManager: responsible for responding to discovery requests from Companion Screens and launching the HbbTV® application.
- CS application: responsible for discovering the HbbTV® terminal (clause 14.7) and requesting the launch of an HbbTV® application (clause 14.6).

For a broadcast independent HbbTV® application to be launched by a CS application it first needs to discover the application Launch service endpoint. Once this has been found, it can then attempt to launch an HbbTV® application on the terminal by providing it with an XMLAIT as the payload of an HTTP POST request to the application Launch service endpoint. There are a variety of reasons why the terminal may refuse the requested application launch, which are described in clause 14.6.2.

14.3 Requirements for launching a CS application from an HbbTV® application

14.3.1 Support for 'launching a CS application from an HbbTV® application'

Terminals shall, in conjunction with an application (provided by the terminal manufacturer or one of their agents) running on a Companion Screen, support the 'Launching a CS application from an HbbTV® application' feature if the terminal manufacturer (or one of their agents) provides a Companion Screen application that can link to, and control the terminal from the Companion Screen application. This Companion Screen application may also be referred to as a 'Launcher application' (see clause 14.3.2).

If this feature is supported, then the following clauses of the present document shall apply:

- From clause 8.2.6, the following APIs are required to be implemented:
  - `discoverCSLaunchers` and its associated callback, `onCSDiscovery;`
  - `launchCSApp` and its associated callback, `onCSLaunch.`
- From clause 14.3.2, all requirements apply, and they shall be implemented by at least one Launcher application from the manufacturer (or its agent):
  - The manufacturer should ensure that all versions of its companion applications that can link to and control a TV support this functionality.
- From clause 14.4, all requirements apply and shall be implemented by the terminal.

If this feature is not supported, the following clauses of the present document shall apply:

- From clause 8.2.6 the `discoverCSLaunchers` API is required to be implemented such that when called it always returns false.

14.3.2 The Launcher application

The protocol between the HbbTV® terminal and the Companion Screen device is not defined by the present document.Where the 'Launching a CS application from an HbbTV® application' feature is supported, a Launcher application is required to be provided and the following behaviours of the Launcher application shall be supported:

- The Launcher application shall be able to connect and communicate with the HbbTV® terminal using the proprietary protocol.
- The Launcher application shall support the following HbbTV® API extensions to the `application/hbbtvCSManager` object defined in clause 8.2.6:
  - `discoverCSLaunchers()` and its associated callback, `onCSDiscovery`
  - `launchCSApp()` and its associated callback, `onCSLaunch`

Regarding the requirement for an HbbTV® terminal manufacturer to provide Launcher application functionality, it is recommended that an HbbTV® terminal manufacturer provides a Companion Screen application with Launcher application functionality for as many Companion Screen platforms and platform variants as possible. It is recommended that a Launcher application can provide optional mechanisms such as the ability to block, auto allow, explicitly request the users permission, or remember a user's previous responses.

For the purposes of this feature and the APIs defined in clause 8.2.6, the following terms are defined:

- Connected - If a CS Launcher application is communicating, or has set-up a communication path (using whatever proprietary means) with an HbbTV® terminal, such that the `launchCSApp()` API would be able to be run and return 'true', then it is said to have 'connected' and be in a 'connected' state. The communication path carries the protocol (not defined by HbbTV®) that enables the "Launching a CS application" feature, and may also carry other information not specified here.

  For a connection to be made, a discovery and/or an association step may be required. How this is achieved is out of scope of the current specification.

  There is no requirement for the connection to stateful or persistent.
- Disconnected - A CS launcher is defined as "disconnected" if it is not "connected". Examples of disconnected states occur if the Launcher application terminates, or if the launcher application connects to another HbbTV® terminal (and is no longer connected to the initial HbbTV® terminal).

It does not necessarily cover the instance of a network infrastructure failure, but it may do if the communication path between the Launcher app and the HbbTV® terminal is connection-oriented (e.g. TCP based).

14.4 Launching a CS application from an HbbTV® application

14.4.1 CS OS identification

14.4.1.1 General (informative)

A Companion Screen OS Identifier (CS OS ID) is a string containing the following information about a Companion Screen Device:

1) the identity of the Launcher application;

2) a list of app store(s) that the Launcher application is aware of;

3) an existing User Agent string representative of a browser engine on the Companion Screen Device.

This is an example CS OS ID string. It has been split over several lines to ease understanding:

```
com.my-oem.tv.applauncher/1.3.7
(appstore/com.android.vending; appstore/com.amazon.venezia)
Mozilla/5.0 (Linux; U; Android 2.3.5; en-gb; HTC Desire HD A9191 Build/GRJ90) AppleWebKit/533.1
(KHTML, like Gecko) Version/4.0 Mobile Safari/533.1
```

In the example the Launcher application is identified as "`com.my-oem.tv.applauncher`" version "`1.3.7`". Two application stores have been identified: "`com.android.vending`" (the google app store) and "`com.amazon.venezia`" (the amazon kindle app store).

The remainder of the example CS OS ID is a User Agent String that indicates the device runs the Android™ operating system and indicates its make and model. It identifies the browser engine used within WebView UI elements in that device's Android™ installation.

Clauses 14.4.1.3 - 14.4.1.4 provide guidance on the generation of the CS OS ID for specific Companion Screen Device OSes.

14.4.1.2 Syntax and semantics

The syntax is expressed as augmented BNF as defined in Table 15 of IETF RFC 5234 [39].

Table 28: Augmented BNF syntax of the Companion Screen OS Identifier string

BNF syntax of the Companion Screen OS Identifier string	
csoid	= launcher WS user_agent_string
launcher	= launcher_product [launcher_comment]
launcher_product	= launcher_name "/" launcher_version
launcher_comment	= WS "(" comment_body ")"
comment_body	= comment 0*(";" WS comment)
comment	= store_info | manufacturer_specific_comment
store_info	= "appstore" "/" app_store_id
WS	= 1*" "
NOTE: launcher_name, launcher_version, app_store_id and manufacturer_specific_comment are defined below.	

The CS OS ID is structured in a format consistent with a User Agent String as defined in IETF RFC 2616 [6]. It consists of a collection of whitespace separated product tokens and comments. Product tokens are structured as a name followed by a slash "/" character followed by a version. Comments are text enclosed by parenthesis and immediately follow the product token that they relate to. If a comment is not needed (because it will contain no information) then it may be omitted.

The first product token and comment in the CS OS ID string consist of a launcher product token for the Launcher application and a launcher comment for the Launcher application. The comment optionally lists available app stores on the Companion Screen Device or any Launcher application developer specific additional information.

The remainder of the CS OS ID string is a User Agent string provided or derivable directly from the platform/operating-system on which the Launcher application is running.

The Launcher product token consists of a launcher name and launcher version that identifies the Launcher application. The name shall be a reverse DNS notation formatted identifier uniquely associated with the application and its developer. The root of the domain name used shall be a domain name registered by the application developer. The version should be a string representation of the version number of the Launcher application.

The launcher comment is made up of zero, one or more whitespace separated tokens. Tokens can include whitespace if they are enclosed with a double quote mark character at beginning and end.

Manufacturer or Launcher application specific tokens can be used, but may not use the prefix "`appstore/`", except for the following purpose: The launcher comment may contain one or more tokens listing application stores available on the Companion Screen Device that the Launcher application is aware of. Each application store token consists of a prefix "`appstore/`" followed by an `app_store_id` in reverse domain name notation.

The presence of app store tokens is optional where the OS of the Companion Screen Devices has a single known fixed app store (such that the TV application can know which app store to use simply by identifying the OS from examination of the User Agent string part). In all other situations, app store tokens should be included. The set of `store_id`s returned may be supplemented by any other `app_store_id`s that the Launcher application is aware of being present on the Companion Screen Device.

The User Agent string for the Companion Screen Device shall be derived from one of the following sources (if obtainable), listed in order of preference (the first being the most preferred):

1) User agent string of the browser application that is set up to be the current default browser on the Companion Screen Device.

2) The default user agent of an embeddable web-view UI element provided by the platform.

3) A system-wide generic default value for a User Agent string.

4) A synthetic User Agent string generated by the Launcher application itself.

Any option from those listed above is considered obtainable if it is possible to obtain that User Agent string reliably using documented public platform-provided APIs and without requiring user interaction or the installation or execution of other application on the Companion Screen Device and without needing to use network connectivity to the internet or other devices.

14.4.1.3 Hints on how to derive the CS OS identifier on Android™ (informative)

The hints described in this clause are considered applicable for all current commonly used releases of Android™ up to and including Android™ 4.3 as of December 2013.

The launcher product token should be based on fields from the Android™ application manifest. The application "package name" should be used as the name part and the "version name" as the version part.

`app_store_id`s should be included for each app store that the Launcher application is aware is available on the device on which it is running. It can determine the list of installed intent handlers for "`market://`" URLs by creating an intent, then querying the activities that will handle it using the system provided package manager. This returns a list of reverse domain name notation package names for the installed packages that will handle requests to open "`market://`" URLs. The code below demonstrates this:

```
Intent market = new Intent(Intent.ACTION_VIEW, Uri.parse("market://search?q=dummy"));
PackageManager manager = getPackageManager();
List<ResolveInfo> list = manager.queryIntentActivities(market, 0);
String comment = "(";
for (ResolveInfo item : list) {
 comment = comment + " appstore/" + item.activityInfo.applicationInfo.packageName;
}
comment = comment + ")";
```

The list of package names should be each treated as a separate `store_id`, and listed in the same order that they feature in the list returned by the API call.

A Launcher application should obtain a User Agent string from source (2) listed in clause 14.4.1.2. A WebView UI element is created and its settings attribute is then queried to obtain the (default) User Agent string being used by the WebView UI element:

```
WebView webview = new WebView(context);
WebSettings settings = webview.getSettings();
String user_agent = settings.getUserAgentString()
```

If that is unavailable then the following code can currently be used to obtain a User Agent string from source (3) listed in clause 14.4.1.2:

```
String user_agent = System.getProperty("http.agent")
```

If a future release of Android™ makes it possible to obtain a user agent string from source (1) listed in clause 14.4.1.2, then Launcher applications that target those versions of Android™ should use that.

14.4.1.4 Hints on how to derive the CS OS identifier on iOS™ (informative)

The hints described in this clause are considered applicable for all current commonly deployed releases of iOS™ up to and including iOS™ 7 as of December 2013.

The launcher product token should be based on fields from the iOS™ application project settings. The application bundle identifier should be used as the name part. Either the build/bundle version or short version string should be used as the version part.

There is a single iTunes application store. This may optionally be identified within the launcher app `comment` field by including the token "`appstore/com.apple.itunes`".

A Launcher application should obtain a User Agent string from source (2) listed in clause 14.4.1.2. A UIWebView UI element is created and JavaScript is executed within it to return the `navigator.userAgent` object:

```
UIWebView* webView = [[UIWebView alloc] init];
NSString* userAgent;
userAgent = [NSString stringWithString:
     [webView stringByEvaluatingJavaScriptFromString:@"navigator.userAgent;"]];
```

An alternative method for obtaining this information is to initiate an HTTP request within a `UIWebView` UI element, but to intercept and block it using the delegate function `shouldStartLoadWithRequest()`. The delegate function can examine the request object to determine the value of the user agent header field. The following source code demonstrates this:

```
/* ========== UserAgentObtainer.h ========== */
@interface UserAgentObtainer : NSObject<UIWebViewDelegate> {
    NSString* userAgent;
}

@property (nonatomic,retain) NSString *userAgent;
-(id)init;
@end

/* ========== UserAgentObtainer.m ========== */
#import "UserAgentObtainer.h"

@implementation UserAgentObtainer
@synthesize userAgent;

-(id)init {
    self = [super init];
    if (self) {
        UIWebView* webView = [[UIWebView alloc] init];
        webView.delegate = self;
        [webView loadRequest:
            [NSURLRequest requestWithURL:
                [NSURL URLWithString:@"http://null.com"]]];

        while (self.userAgent==nil) {
            [[NSRunLoop currentRunLoop]
                runMode:NSDefaultRunLoopMode
                beforeDate:[NSDate distantFuture]];
        }
    }
    return self;
```

```
}

-(BOOL)webView:(UIWebView *)webView
    shouldStartLoadWithRequest:(NSURLRequest *)request
    navigationType:(UIWebViewNavigationType)navigationType
{
    self.userAgent=[request valueForHTTPHeaderField:@"User-Agent"];
    return NO;
}

@end
```

If a future release of iOS™ makes it possible to obtain a user agent string from source (1) listed in clause 14.4.1.2, then Launcher applications that target those versions of iOS™ should use that.

14.4.2 Payload format for Install and Launch operations

14.4.2.1 Permissible Operations

There are two operation instructions that may be carried in the payload data of the `launchCSapp()` JS API. These are:

- Install application (for Native applications only)
- Launch application (for either Native or HTML applications)

The format of these instructions is described in clause 14.4.2.2. Either one or two of these operation instructions may appear in the payload data with the following possible combinations:

- Launch Only

  If the Launch Native or Launch HTML instruction is supplied, then the Launcher application shall attempt to launch the application. If the launch fails for any reason, then the `launchCSapp` may (using the `onCSLaunch` callback) respond with an appropriate error code as described in clause 8.2.6.1.

- Install Only

  If an Install (Native) application instruction is supplied, then the Launcher application shall attempt to install the native application using the store information (if present) and the store specific location information. If no store information is provided a platform default store shall be used. If the install fails for any reason, then the `launchCSapp` may (using the `onCSLaunch` callback) respond with an appropriate error code as described in clause 8.2.6.1. See clause 14.4.2.2.1 for more information on the install operation.

- Both Launch Native and Launch HTML

  If both a Launch Native and a Launch HTML instruction are supplied the Launch Native application instruction shall be actioned first. If this is successful, then the launch of the HTML application shall not be executed. If the Launch Native application is not successful, then the launch of the HTML application shall be actioned. No response should be made to the HbbTV® application at this stage using the `onCSLaunch` callback. If the launch of the HTML application is not successful, then the `launchCSapp` may (using the `onCSLaunch` callback) respond with an appropriate error code as described in clause 8.2.6.1.

  In this combination, the Launch Native instruction shall be the first instruction in the payload and the Launch HTML instruction the last, otherwise the combination is not valid with the Launcher application behaviour undefined.

Other operation combinations are invalid. In particular Install Native and Launch Native are not a valid combination of operations; separate operations have to be used to achieve this.

14.4.2.2 JSON payload format

14.4.2.2.1 Introduction

The payload data for the install and launch operations are carried as strings which contain JSON formatted data. The exact format of the JSON payload is described here. JSON not conforming to these rules (including empty strings) are invalid and shall cause the `onCSLaunch` callback to return with an error code of 4 (general_error).

The terminal and Launcher application shall support a maximum size for the string containing the JSON formatted data of 65 536 bytes.

The JSON schema [i.10] for the launch and install operation payload formats are described in clause 14.4.2.2.4.

14.4.2.2.2 Install operation

This Installation Operation may contain multiple sources for an installation to be sourced from, but shall include at least one source.

If there are multiple sources, then each source shall contain a store name, except for the last one in the list, which may optionally contain a store name. If the last one in the list does not contain a store name, the platform default store shall be assumed.

If there are multiple sources then the Launcher application shall attempt to install from the first store in the list that the platform recognises, unless there is a specific reason for the Launcher application to do otherwise. Valid specific reasons such as the following may exist:

- a user has set a preference for a particular store to be used;
- the Launcher application or Companion Screen platform can automatically select a store that provides the end user with the best offer.

If there is a single source, then it may contain a store name. If it does not, the platform default store shall be assumed.

The store names (`appStoreId`) are defined as the `app_store_id` string as defined in Table 28 in clause 14.4.1.2.

In all cases the install URL shall be appropriate to the associated store.

An example for Android is as follows, where the application can be installed from either the Google Play store or Amazon App stores (specified using `market://` and `amzn://` URLs respectively):

```
{
  "install" : [
    {
      "installUrl" : "amzn://apps/android?p=com.examples-r-us.games.puzzle_game",
      "appStoreId"  : "com.amazon.venezia"
    },
    {
      "installUrl" : "market://details?id=com.examples-r-us.games.game-of-speed"
    }
  ]
}
```

An example for iOS is as follows, where the application is to be installed from the Apple app store (specified using an `itunes.apple.com` URL):

```
{
  "install" : [
    {
      "installUrl" : "https://itunes.apple.com/app/subway-surfers/id512939461?mt=8"
    }
  ]
}
```

14.4.2.2.3 Launch operation

The Launch Operation shall contain either a single launch entry or two launch entries. Zero, or more than two entries are invalid.

If there is a single launch entry then the type shall be either "`native`" or "`html`" as shown below.

If there are two launch entries, then the first launch entry shall be of type "`native`", and the second shall be of type "`html`", as shown below.

An example for a single launch entry (of type "`html`") is shown as follows:

```
{
```

```
  "launch" : [
    {"launchUrl" : "https://www.examples-r-us.com/great-game.html", "appType" : "html"}
  ]
}
```

An example for a two launch entry is shown as follows:

```
{
  "launch" : [
    {"launchUrl" : "g-quiz://com.examples-r-us.games.quiz-
game?colour=blue&app2app_uri=ws://192.168.1.11:992/hbbtv/", "appType" : "native"},
    {"launchUrl" : "https://www.examples-r-us.com/quiz-fallback-app.html?
colour=blue&app2app_uri=ws://192.168.1.11:992/hbbtv/", "appType" : "html"}
  ]
}
```

14.4.2.2.4 JSON payload schema

The JSON schema for the install and launch operations payload of the `launchCSapp()` function is defined as follows:

```
{
  "id": "http://hbbtv.org/cs-install#",
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "HbbTV CS Application Install",
  "description": "HbbTV CS Application Install Schema as defined in HbbTV 2.0 (TS 102 796
v1.4.1), clause 14.4.2. (c) 2014 hbbtv.org - All rights reserved.",
  "type": "object",
  "oneOf": [
    {
      "type": "object",
      "required": [
        "install"
      ],
      "install": {
        "type": "array",
        "items": {
          "type": "object",
          "properties": {
            "installUrl": {
              "type": "string",
              "format": "uri"
            },
            "appStoreId": {
              "type": "string"
            }
          },
          "required": [
            "installUrl"
          ],
          "additionalProperties": "false"
        },
        "minItems": 1
      }
    },
    {
      "type": "object",
      "required": [
        "launch"
      ],
      "launch": {
        "type": "array",
        "items": {
          "type": "object",
          "properties": {
            "launchUrl": {
              "type": "string",
              "format": "uri"
            },
            "appType": {
              "enum": [
                "native",
                "html"
              ]
            }
          },
          "required": [
            "launchUrl",
            "appType"
```

```
            ],
            "additionalProperties": "false"
          },
          "minItems": 1,
          "maxItems": 2
        }
      }
    ]
  }
```

14.4.2.2.5 Handling Special Characters in URLs (Informative)

application authors should take care to avoid the use of, or correctly escape, any special characters in URL parameters. In particular, authors should note that the parameters in any URLs are carried in JSON data defined by the schema in clause 14.4.2.2.4 and that any such JSON passed via the `launchCSApp()` API is expected to be valid JSON. Furthermore, application authors may also need to consider any CS OS specific restrictions and the carriage of URLs via CS OS APIs used to launch the Natice CS applications

14.5 Application to application communications

14.5.1 General

A terminal shall provide an application-to-application communication service as described here to enable an HbbTV® application to communicate concurrently with one or more Companion Screen applications and/or applications running on other HbbTV® terminals present on devices on the same home network as the terminal.

NOTE 1: The identity of the other party with which an application is communicating is not authenticated by the application to application communication protocol and the integrity of the messages exchanged also cannot be assumed. application developers are strongly recommended to consider these factors when designing the protocols to be tunnelled within the application to application communication protocol and also when implementing code that processes received messages.

NOTE 2: Application to application communication is also not generally suitable to be used to communicate credit card details, PIN numbers or other sensitive data. Application developers are free to implement their own security protocols tunnelled within the application to application communication protocol to encrypt this data, however this is generally not recommended. For sensitive data it is more appropriate to relay this to internet servers using an established and well supported secure communications protocol such as HTTPS.

The terminal shall implement a server providing endpoints, described in clause 14.5.2, that implement the server-side of the Websocket protocol version 13 as defined in IETF RFC 6455 [40]. The server shall be able to accept connections once an HbbTV® application has called the getApp2AppLocalBaseURL() method and until the application exits. The server may be able to accept connections at other times but this is implementation dependent and outside the scope of the present document. If the server is not able to accept connections then the server shall either abort the opening WebSocket handshake as described in clause 7.2.2 of IETF RFC 6455 [40] or simply not have the TCP port open at all.

HbbTV® applications determine the location of the service endpoints using JavaScript APIs defined in clause 8.2.6. HbbTV® applications and Companion Screen applications connect to the service endpoints using the WebSocket protocol in the role of a client of the WebSocket protocol. The terminal shall handle connection requests from clients (HbbTV® applications or Companion Screen applications) in the manner defined in clause 14.5.4 and apply pairing rules defined in clause 14.5.5 to determine whether to pair connections from two clients. It shall then act as a relay, as defined in clause 14.5.6 to relay messages between the two client connections that are paired.

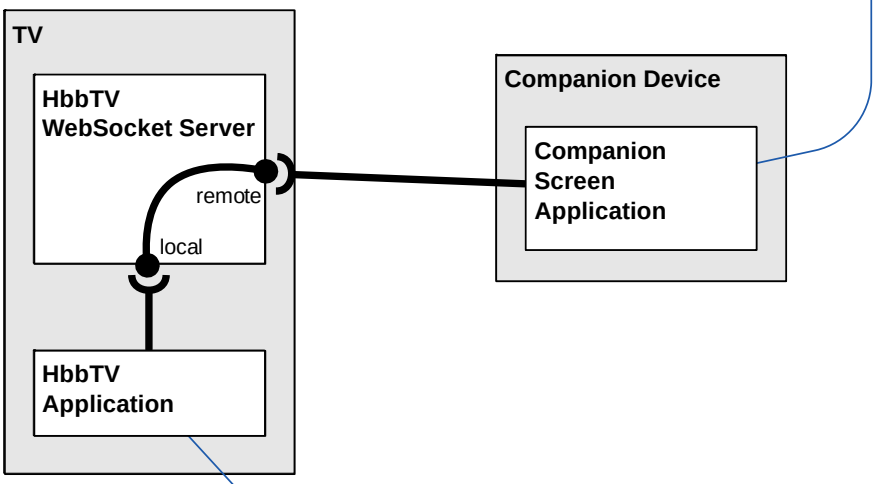
EXAMPLE: Figure 34 illustrates the application to application communication service in use. An HbbTV® application and a Companion Screen application use the WebSockets API, as defined by the W3C Websocket API Recommendation [41] to create WebSocket connections. The connections are then paired by the terminal, meaning that it will relay messages between the two clients through the WebSockets protocol connections.

```
app2appRemoteBaseUrl = <<obtained during discovery of terminal on the home network>>
appEndpoint = “org.mychannel.myapp”;

ws = new WebSocket(app2appRemoteBaseUrl + appEndpoint);

ws.onopen = function(evt) { alert(“Connection waiting ...”); };
ws.onclose = function(evt) { alert("Connection closed."); };

ws.onmessage = function(evt) {
    if (evt.data == “pairingcompleted”) {
        alert(“Connection paired.”);
        ws.onmessage = function(evt) { alert( "Received Message: " + evt.data); };
    } else {
        alert("Unexpected message received from terminal.");
        ws.close();
    }
}
```




```
app2appLocalBaseUrl = hbbtvCSManagerInstance.getApp2AppLocalBaseURL();
appEndpoint = “org.mychannel.myapp”;

ws = new WebSocket(app2appLocalBaseUrl + appEndpoint);

ws.onopen = function(evt) { alert(“Connection waiting ...”); };
ws.onclose = function(evt) { alert("Connection closed."); };

ws.onmessage = function(evt) {
    if (evt.data == “pairingcompleted”) {
        alert(“Connection paired.”);
        ws.send("Hello WebSockets!");
    } else {
        alert("Unexpected message received from terminal.");
        ws.close();
    }
}
```

Figure 34: Application to application communication using WebSockets

14.5.2 Service endpoints provided by the terminal

The terminal shall provide two service endpoints implementing the server-side of the WebSocket protocol specification IETF RFC 6455 [40]:

- The local endpoint is for connecting to by clients that are HbbTV® applications on the terminal.
- The remote endpoint is for connecting to by clients that are applications on other devices on the home network, including remote Companion Screen applications or applications running on other HbbTV® terminal devices.

HbbTV® applications only connect to the local service endpoint of the terminal on which they are running, or to a remote service endpoint of a different terminal on the same home network. Companion Screen applications and applications on other HbbTV® terminals only connect to the remote service endpoint of any terminal.

It is recommended that the terminal should not make it possible to connect to the local service endpoint from other devices within the home network.

NOTE: This can be achieved, for example, by locating the local service endpoint only on a local loopback interface within the terminal.

Both endpoints shall satisfy the security requirements of clause 11.7 of the present document.

14.5.3 Handling of new connections from clients

The terminal shall support a minimum of 10 concurrent WebSocket connections to the local service endpoint from local HbbTV® applications and, simultaneously, a minimum of 10 concurrent WebSocket connections to the remote service endpoint from other terminals or companion screen applications.

The terminal shall reject requests to the service endpoints if it cannot handle more concurrent connections. Otherwise, it shall accept the WebSocket connection and complete the WebSocket protocol handshake.

The client, however, waits until pairing is completed (according to the rules defined in clause 14.5.4) before sending data frames to be relayed. A connection in this state constitutes a waiting connection. The terminal informs the client of successful pairing as defined in clause 14.5.5. In case the terminal wishes to implement a time-out, it shall send a Close frame (as defined in the WebSocket protocol specification clause 5.5.1, IETF RFC 6455 [40]).

If the resource-name used in the GET line of the request handshake from the client does not match the rules defined in clause 14.5.4 for the application to application service endpoint, the terminal shall respond with a 404 Not Found response and close the WebSocket connection.

The terminal shall ignore any `Origin` header in the request handshake sent by the client.

Terminals are not required to support the `Sec-WebSocket-Protocol` header defined in IETF RFC 6455 [40], clause 11.3.4.

Terminals shall not use any WebSocket extensions. Terminals shall ignore any `Sec-WebSocket-Extensions` header in the request handshake sent by the client. Terminals shall not send a `Sec-WebSocket-Extensions` reply header.

EXAMPLE: Figure 35 illustrates the situation where a HbbTV® application, acting as a client, has made a connection to the local service endpoint with a base-url-resource-name of "`/hbbtv/`" with an `app-endpoint` of "`org.mychannel.myapp`". The connection is now in a waiting state because no Companion Screen application has yet connected to the application to application communication service using the same `app-endpoint`.

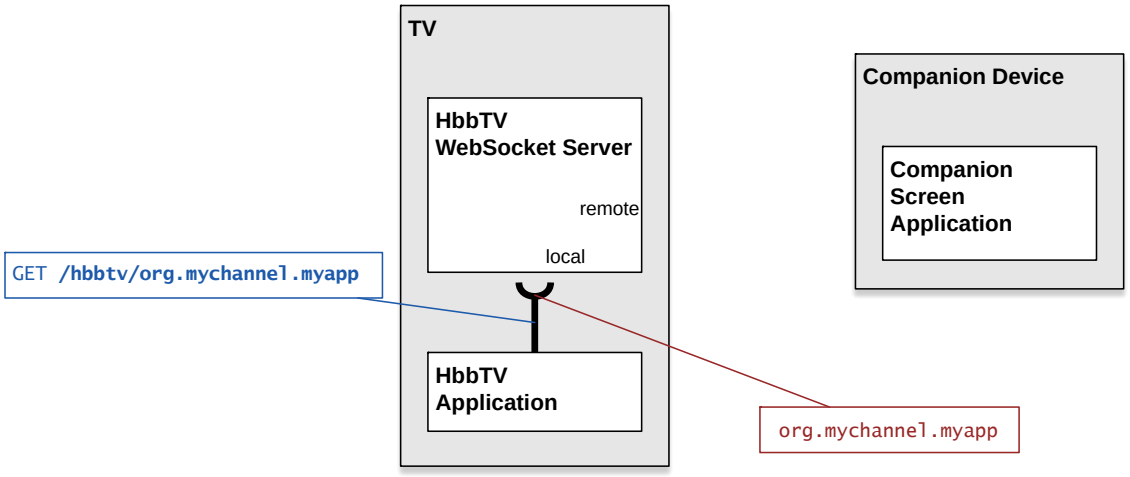


Figure 35: A waiting connection for application to application communication

NOTE: Clients are not advised to attempt to request another connection to a service endpoint before any existing waiting connection to that service endpoint was either successfully connected or has timed-out or otherwise been disconnected.

If an HbbTV® application wishes to communicate to more than one Companion Screen application, it can do so by waiting until an existing waiting connection has become paired, and then issue a further connection requests to the service endpoint and repeat this until the maximum number of client-to-client connections the terminal is able to process has been reached. This is illustrated in figure Figure 36.

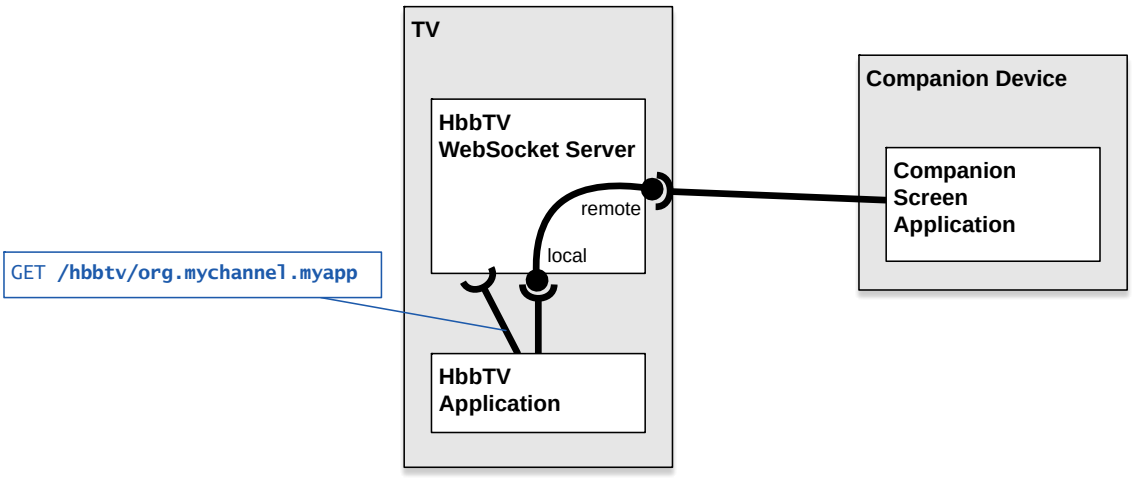


Figure 36: A paired connection for application to application communication

14.5.4 Connection pairing

A WebSocket URL, as defined in clause 3 of IETF RFC 6455 [40], defines the host, port, security, and resource-name of a service endpoint that supports the WebSocket protocol. The terminal provides WebSocket URLs, known as the base WebSocket URLs, for each of the local and remote service endpoints.

The WebSocket URL for the remote service endpoint that is provided by the discovery mechanism shall use the "`ws:`" scheme.

NOTE 1: The secure mode of WebSockets cannot be used because certificate authorities will not issue certificates for a server having a dynamic or private IP address. Such a server could not present a suitable certificate chain. For more information, see clause 7.1.4.2.1 of the CA/Browser Forum Baseline Requirements [i.17]. See also clause A.3.13 “Mixed Content”.

NOTE 2: These base WebSocket URLs are retrieved by an application from the terminal via the `getApp2AppLocalBaseURL()` and `getApp2AppRemoteBaseURL()` methods defined in clause 8.2.6.1. The base WebSocket URL for the remote service endpoint is also advertised through the terminal service endpoint discovery mechanism described in clause 14.7.2.

The WebSocket URL that a client connects to when using either service endpoint is formed by concatenating the base WebSocket URL for that service endpoint with an application specific suffix. This suffix is referred to in the present document as the app-endpoint.

To establish a WebSocket connection, clients first establish a TCP connection to the host, and port as specified by the base WebSocket URL. Then, in the opening handshake of the protocol as defined in IETF RFC 6455 [40], the client specifies a resource name that comprises the resource name from the base WebSocket URL concatenated with the app-endpoint.

EXAMPLE 1: A terminal advertises its remote endpoint for application to application communication as the base WebSocket URL "`ws://192.168.1.5:8140/hbbtv/`". A Companion Screen application wishes to use this service endpoint with an app-endpoint of "`uk.co.bbc.cs-svc`". The Companion Screen application therefore uses the W3C WebSocket API [41] to request a WebSocket connection be established to the WebSocket URL "`ws://192.168.1.5:8140/hbbtv/uk.co.bbc.cs-svc`":

ws = new WebSocket("ws://192.168.1.5:8140/hbbtv/uk.co.bbc.cs-svc");

The first line of the opening handshake sent by the API implementation is:

```
GET /hbbtv/uk.co.bbc.cs-svc HTTP/1.1
```

`app-endpoint` is application specific. This is used in the process of pairing this connection with another connection from the other service endpoint. It will be chosen by developers to avoid collisions with other developers' applications.

NOTE 3: Developers can avoid collisions by using, for example, a reverse DNS notation formatted identifier uniquely associated with the HbbTV® application or Companion Screen application and its developer. Another possible option is to use an assigned HbbTV® organisation id and application id. This could be formatted as organisation id followed by a period "." character followed by an application id, where the ids are written as hex digits.

The terminal shall support an `app-endpoint` of any length from 1 to at least 1 000 characters in length and which contains any characters permitted in a `resource-name` by IETF RFC 6455 [40].

The terminal shall pair two waiting connections according to the following rules:

- One waiting connection shall be on the local service endpoint (and therefore be inferred to have come from the HbbTV® application client).
- One other waiting connection shall be on the remote service endpoint (and therefore be inferred to have come from a remote client, such as a Companion Screen application).
- The `app-endpoint portion` of the resource name used in the client handshake request shall match between both waiting connections.

While there is a waiting connection on the local service endpoint, the terminal shall apply these rules to determine whether there are two waiting connections that can be paired. If there is more than one waiting connection on the remote service endpoint that could be paired with the waiting connection on the local service endpoint, the terminal shall select only one of them for pairing with the waiting connection on the local service endpoint. The terminal shall keep the remaining connections in the waiting state. Later, when new connections are made (to either endpoint), these rules are re-evaluated to try to create more pairings.

NOTE 4: No rules are defined in the present document for how the terminal decides which waiting connection on the remote service endpoint is selected for pairing when multiple ones are available. Developers cannot assume any particular algorithm is employed (such as selecting the one that has been waiting the longest).

After pairing waiting connections, the terminal shall proceed to provide application to application communication to communication through those connections, as defined in clause 14.5.5.

EXAMPLE 2: Figure 37 illustrates the situation where a HbbTV® application, acting as a client, has made a connection to the local service endpoint with a `base-url-resource-name` of "`/hbbtv/`" and an `app-endpoint` of "`org.mychannel.myapp`". A Companion Screen application has also made a connection to the remote service endpoint with a `base-url-resource` name of "`/hbbtv/`" and the same `app-endpoint` as the HbbTV® application's connection. These two waiting connections will be paired as they satisfy the above rules.

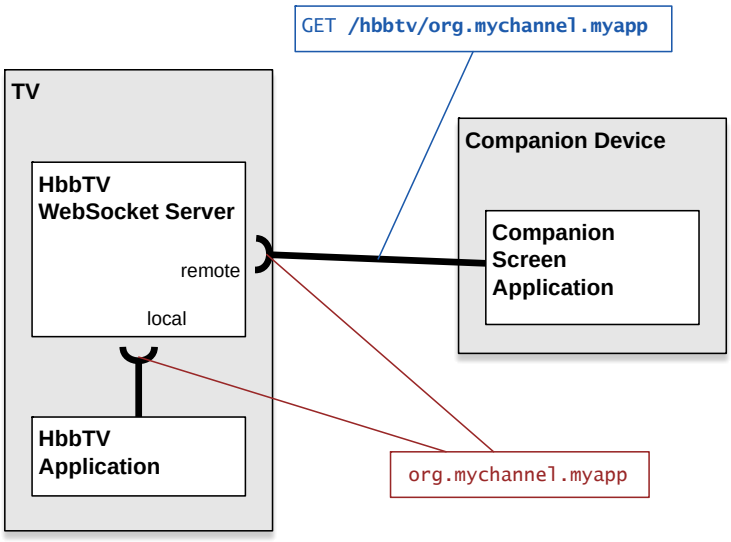


Figure 37: Two waiting connections that will be paired by the TV

14.5.5 Paired connections

When connections from two clients enter into a state of being paired to each other, the terminal shall immediately inform both clients by sending them a Data frame of type Text (as defined by the WebSocket protocol specification clause 5.6 in IETF RFC 6455 [40]) with as Payload data the UTF-8 encoded text '`pairingcompleted`'. The connections are now both considered to be open, and the clients to be paired.

Once paired and connections to both clients are open, the terminal shall act as a relay to pass messages between them, providing, in effect, a full-duplex bi-directional communication stream. When either client sends a WebSocket message, consisting of one or more protocol frames, the terminal, upon receipt of each frame, shall immediately relay its contents to the other client via the corresponding WebSocket connection and maintaining the same payload type.

The terminal shall discard any data frames received from a client before it has informed that client of successful pairing and shall relay all data frames thereafter. Additionally the terminal shall inform a client of successful pairing before sending it relayed data frames.

The terminal shall support all data frame types and both unfragmented and fragmented frames as required by IETF RFC 6455 [40].

When relaying a payload received from a client, the terminal is not required to fragment the payload across frames in the same way as the frames it received.

The terminal shall be able to handle relay messages with a payload size up to and including 131 072 bytes. For messages sent from the remote client, the terminal shall be able to relay the message, and have it received by the local application (client), if it is fragmented where each frame may carry any number of bytes up to the size of the message.

Over a 10 second period, during which any other paired connections have no traffic, the terminal shall be able to relay any of the following rates of traffic across a single paired connection:

- 10 messages with a payload size of up to and including 131 072 bytes sent by the client connected to the local service endpoint.
- 10 unfragmented messages with a payload size of up to and including 131 072 bytes sent by the client connected to the remote service endpoint.

- 200 messages with a payload size of up to and including 512 bytes sent by the client connected to the local service endpoint.
- 200 unfragmented messages with a payload size of up to and including 512 bytes sent by the client connected to the remote service endpoint.

When messages sent by all clients across all currently paired connections are considered in aggregate then, during a 10 second period, the terminal shall be able to relay any of the following rates of traffic when spread evenly across up to 10 paired connections:

- 50 messages with a payload size of up to and including 131 072 bytes sent by the application connected to the local service endpoint (5 frames via each paired connection).
- 50 unfragmented messages with a payload size of up to and including 131 072 bytes sent by the client connected to the remote service endpoint (5 frames via each paired connection).
- 250 messages with a payload size of up to and including 512 bytes sent by the client connected to the local service endpoint (25 frames via each paired connection).
- 250 unfragmented messages with a payload size of up to and including 512 bytes sent by the client connected to the remote service endpoint (25 frames via each paired connection).

If the client connected to the remote service endpoint sends a Ping frame(as defined in IETF RFC 6455 [40]) then the terminal shall respond with a Pong frame.

If the application closes the WebSocket connection to the local service endpoint then the terminal shall commence the process of disconnecting the corresponding paired connection from the remote other client by sending a corresponding Close frame as defined in IETF RFC 6455 [40]. If the application is stopped and WebSocket connections are still open, then any WebSocket connections to the WebSocket server shall be closed in an undefined manner.

If the remote client sends a Close frame as defined in IETF RFC 6455 [40] or disconnects without sending a Close frame, the terminal shall commence the process of disconnecting the client. In addition, it shall close the corresponding paired WebSocket connection that was made by the application to the local service endpoint.

In normal operation the terminal should indefinitely maintain the pair of connections and relay messages as described above. However, if the terminal has initiated the closure of the connection to either client, then it shall close both connections in the pair.

14.6 Launching an HbbTV® application from a CS application

14.6.1 Introduction

This clause introduces the methods to launch a broadcast independent HbbTV® application on an HbbTV® terminal from a Companion Screen application.

It consists of the following steps:

- first, the Companion Screen application discovers available DIAL servers;
- then for each DIAL server, it discovers the location of its DIAL REST service.

Then, optionally, the Companion Screen application checks that the HbbTV® DIAL application is supported by the DIAL server, which means that the DIAL server is implemented in a terminal supporting the application launch feature. Finally, using the DIAL application Resource URL for HbbTV® (derived as defined in clause 14.7.2), it attempts to launch the HbbTV® application. These steps are detailed in clause 14.6.2.

14.6.2 Launching an HbbTV® application protocol

The protocol for launching an HbbTV® application from a Companion Screen application is described in this clause.

The Companion Screen requests the launch of the HbbTV® application using the mechanisms defined in clause 6.2.1 of DIAL [50]. It is done by sending an HTTP POST request to the DIAL REST Service URL with the identifier "HbbTV"

for the application, as registered with the DIAL registry [i.8]. The DIAL REST Service URL is obtained from the discovery phase using DIAL Service Discovery (see clause 14.7 and clause 5 of [50]). The BODY data of the HTTP POST request shall contain an XML AIT describing the HbbTV® application to be launched (see clause 7.2.3.2).

An example XML AIT is given below:

```
<?xml version="1.0" encoding="UTF-8"?>
<mhp:ServiceDiscovery xmlns:mhp="urn:dvb:mhp:2009"
                      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <mhp:ApplicationDiscovery DomainName="example.com">
    <mhp:ApplicationList>
      <mhp:Application>
        <mhp:appName Language="eng">Whizzo Play Along Quiz</mhp:appName>
        <mhp:applicationIdentifier>
          <mhp:orgId>123</mhp:orgId>
          <mhp:appId>456</mhp:appId>
        </mhp:applicationIdentifier>
        <mhp:applicationDescriptor>
          <mhp:type>
            <mhp:OtherApp>application/vnd.hbbtv.xhtml+xml</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>01</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>
        </mhp:applicationDescriptor>
        <mhp:applicationTransport xsi:type="mhp:HTTPTransportType">
          <mhp:URLBase>http://www.example.com/whizzo-app.html</mhp:URLBase>
        </mhp:applicationTransport>
        <mhp:applicationLocation>?launch=from-cs</mhp:applicationLocation>
      </mhp:Application>
    </mhp:ApplicationList>
  </mhp:ApplicationDiscovery>
</mhp:ServiceDiscovery>
```

On receiving the HTTP POST request, the terminal shall attempt to launch the HbbTV® application.

If the launch succeeds then the terminal shall respond with the response code 201.

If the launch could not be completed because the application could not be retrieved successfully then the terminal shall respond with the response code 404.

The terminal might have states where the feature is temporarily unavailable, e.g. during a channel scan. The states when the feature is not available are not defined by the present document. If the terminal rejects the application launch for this reason it shall respond with the response code 503.

Terminals shall not, by default, launch applications without at least one of the following approvals or pre-approvals:

- Explicit approval by the user to launch the application at the time the launch request is made.

NOTE 1: The mechanism by which approval is requested needs to be comprehensible to users who are not technologically aware and secure against malevolent applications or devices on the home network. One example of such would be to assume that the user explicitly requested an application on a companion screen to in turn request the HbbTV® application to be launched. Hence a terminal UI could ask the user if they just requested an HbbTV® application be launched. This would avoid any need to identify the application with information that is not, itself, secured and not very comprehensible.

- Explicit pre-approval by the user that the specific application can be launched (for example by the mechanism referred to above).
- Explicit pre-approval by the manufacturer.
- Explicit pre-approval by another party managing the network or market where the terminal is located.

In cases of pre-approval, at the time of approval, the `<applicationTransport>` and `<applicationLocation>` elements from the XML AIT shall be stored.

At the time launching is requested, the terminal shall determine if an application is pre-approved by comparing the complete `<applicationTransport>` element and of that part of the `<applicationLocation>` element excluding any query or fragment from the request to launch an application with the set of pre-approved values. If a match is found for both of these then the application shall be considered pre-approved regardless of mismatches in other values from the XML AIT.

If a requested application is not pre-approved then terminals that support explicit approval by the user to launch the application at the time the launch request is made shall ask the user for that explicit approval.

NOTE 2: For HbbTV® terminals which only use pre-approval, it may be necessary for some applications to be pre-approved (and others not to be pre-approved) in order to run the HbbTV® test suite.

The terminal UI should provide means for the user to either approve or pre-approve application launching. This may include means for the user to accept or block requests from particular companion devices. If the terminal rejects the application launch because approval or pre-approval by the user was requested and denied, then it shall respond with the response code 403, where the body of the response is the 4 character string "`USER`" and has content type "`text/plain`".

If the terminal rejects the request for reasons other than any of the above, then it shall respond with the response code 403, with an empty response body.

Table 29 summarises the HTTP responses described above.

Table 29: HTTP response codes for application launch requests

Response Code	Response Body (defined for 403 response code only)	Description
201 CREATED		The HbbTV® application was launched successfully.
403 FORBIDDEN	USER	The HbbTV® application could not be launched because of a user action or a user setting.
403 FORBIDDEN		The HbbTV® application could not be launched because the operation is rejected by the terminal.
404 NOT FOUND		The HbbTV® application could not be launched because it could not be retrieved successfully, e.g. due to invalid application URL or application server unavailable.
500 INTERNAL SERVER ERROR		The HbbTV® application could not be launched for a reason other than those described by the other response codes listed in this table. Possible reasons include a malformed or otherwise invalid XMLAIT or an invalid HTML document.
503 SERVICE UNAVAILABLE		The HbbTV® application could not be launched because of the terminal's current state.

The HbbTV® application shall be deemed to have launched successfully when the current document readiness of the Document object of the application transitions from "loading" to the next state.

NOTE 3: A `Document` object's `readyState` attribute returns "`loading`" while the `Document` is loading, "`interactive`" once it is finished parsing but still loading sub-resources, and "`complete`" once it has loaded. The `readystatechange` event fires on the `Document` object when this value changes.

14.6.3 Providing HbbTV® user agent

The Companion Screen can determine the value of the `User-Agent` header that is supplied by the terminal on behalf of an HbbTV® application by sending an HTTP GET request to the DIAL application Resource for HbbTV®, as described in clause 14.7.2. The HTTP response contains an `<X_HbbTV_UserAgent>` element which carries the value of the HbbTV® terminal's `User-Agent` header.

NOTE: By obtaining the HbbTV® terminal's `User-Agent` header value, the Companion application can determine if the HbbTV® terminal provides capabilities needed by the HbbTV® application prior to deciding whether to launch an HbbTV® application. It also enables a Companion application to provide an XML AIT that is customised to the capabilities of the terminal.

14.7 Discovering terminals and their service endpoints

14.7.1 Introduction

In the situation where a CS application has been launched by an HbbTV® application, information regarding the location of the service endpoints exposed by the terminal may be conveyed as parameters in the launch URL, as described in clause 14.4.2.2.3.

However, if the CS application has launched the HbbTV® application, or has been launched independently of the HbbTV® application, then it needs to be able to discover the locations of the service endpoints by some other means. The methods for achieving this are described in clause 14.7.2.

These methods may also be used by another terminal which is running an application that wishes to synchronise content presentation or communicate with the running application on this terminal.

14.7.2 Terminal and service endpoint discovery

HbbTV® is a DIAL [50] application registered at the DIAL registry [i.8]. The registered name for HbbTV® applications is '`HbbTV`'. For terminal and service endpoint discovery, the terminal shall support DIAL [50].

Before the HbbTV® service endpoints can be determined, the DIAL REST Service and the DIAL application Resource URL need to be found. This is achieved using the mechanisms described in DIAL [50], clause 5. This consists of an SSDP M-SEARCH request and response, followed by an HTTP GET to the URL obtained from the LOCATION: header in the M-SEARCH response. This HTTP response contains an Application-URL header and a body. The response body is a UPnP device description as required by clause 5.4 of DIAL [50]. The Application-URL header provides the DIAL REST Service URL.The DIAL application Resource URL for HbbTV® is the DIAL REST Service URL followed by a single slash character ('/') and the application name '`HbbTV`'. For example:

`http://192.168.1.11:11111/apps/HbbTV`

The terminal shall support the HbbTV® DIAL application, and shall respond to an HTTP GET request to the DIAL application Resource URL for HbbTV® with a 200 OK response. The response shall include an XML document, as described in clause 6.1.2 and annex A of DIAL [50], in the body.

NOTE 1: HbbTV® terminals are only required to support discovery of other HbbTV® terminals when they have the capability to act as a slave terminal as defined in clause 10.2.9.3.

The XML document in the HTTP response shall include the mandatory elements and attributes defined in clause 6.1.2 and annex A of DIAL [50] There shall be one `<additionalData>` element containing one of each of the following elements:

- An `<X_HbbTV_App2AppURL>` element that provides the absolute URL of an application to application communication service endpoint, i.e. a Web Socket Server URL, as defined in clause 14.5 and the W3C Web Socket protocol specification IETF RFC 6455 [40].
- An `<X_HbbTV_InterDevSyncURL>` element that provides the absolute URL of a CSS-CII service endpoint, i.e. a URL, as defined in clause 13.6 that is used for inter-device synchronization.
- An `<X_HbbTV_UserAgent>` element that provides the value of the HbbTV® terminal's `User-Agent` header as defined in clause 7.3.2.4. See also clause 14.6.3.

NOTE 2: The present document interprets the DIAL specification [50] schema for the `additionalData` element to be interpreted as per 6.3.2 of [50], i.e. in the application resource XML schema, the line `<xs:any minOccurs="0" processContents="lax"/>` is changed to `<xs:any minOccurs="0" maxOccurs="unbounded" processContents="lax"/>`.

The `xmlns` attribute for the HbbTV® elements defined above shall be present, and shall be set to:

```
urn:hbbtv:HbbTVCompanionScreen:2014
```

The additional elements carried in the `<additionalData>` element shall be encoded using the following XML Schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns="urn:hbbtv:HbbTVCompanionScreen:2014"
   targetNamespace="urn:hbbtv:HbbTVCompanionScreen:2014"
   elementFormDefault="qualified">
   <xs:element name="X_HbbTV_App2AppURL"      type="xs:anyURI"/>
   <xs:element name="X_HbbTV_InterDevSyncURL" type="xs:anyURI"/>
   <xs:element name="X_HbbTV_UserAgent"       type="xs:string"/>
</xs:schema>
```

Implementation Note (Informative)

Clause 6.3 of DIAL [50] indicates that the First-screen application and the DIAL REST Service communicate the location of an `additionalDataURL` for the First-screen application to provide `additionalData` to. It is out of scope of HbbTV® to define how the DIAL REST Service obtains the `additonalData` which populates the `<additionalData>` element of the XML document carried by the HTTP GET response. In the case of HbbTV®, the First-screen application is the HbbTV® environment provided by the manufacturer, as is the DIAL REST service, so the co-ordination of this information is entirely under the control of the terminal manufacturer.

14.7.3 Discovery example (informative)

14.7.3.1 DIAL Service Discovery

This is as per DIAL Service Discovery - there are no additional aspects required for HbbTV®. See the example messages B.1 to B.4 in annex B of DIAL [50].

DIAL Device Discovery Request

A device on a home network initiates device discovery by performing an M-SEARCH from the SSDP protocol with the Search Target header (ST) as defined by DIAL:

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: 2
ST: urn:dial-multiscreen-org:service:dial:1
```

Discovery Response

The terminal responses with HTTP/1.1 OK and LOCATION header, and DIAL ST:

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age = 1800
EXT:
LOCATION: http://192.168.1.11:50201/dial.xml
SERVER: Linux/2.6 UPnP/1.1 Sony-BDP/2.0ST: urn:dial-multiscreen-org:service:dial:1
USN: uuid:00000004-0000-1010-8000-d8d43c1923dc::urn:dial-multiscreen-org:service:dial:1
```

Device Description Request

The home network device requests the device description file by an HTTP GET request to the LOCATION URL:

```
GET /dial.xml HTTP/1.1
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.3; SGP312 Build/10.4.B.0.577)
Host: 192.168.1.11:50201
Origin: http://cs.services.broadcaster.com/
```

Device Description Response

The terminal responds with HTTP/1.1 OK header containing the Application-URL as defined in DIAL:

- Header

```
HTTP/1.1 200 OK
CONTENT-LANGUAGE: <language used in description>
CONTENT-LENGTH: <bytes in body>
CONTENT-TYPE: text/xml; charset="utf-8"
Application-URL: http://192.168.1.11:11111/apps
Access-Control-Allow-Origin:*
```

14.7.3.2 DIAL Rest Service

As, from the Device Description Response example, the DIAL REST service is on an `Application-URL` of `http://192.168.1.11:11111/apps` then the following are examples of how the HbbTV® service endpoints and the `User-Agent` header value are discovered.

Application information request

A HTTP GET message is sent to 192.168.1.11, port 11111 as follows:

```
GET /apps/HbbTV HTTP/1.1
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.3; SGP312 Build/10.4.B.0.577)
Host: 192.168.1.11:11111
Origin: http://cs.services.broadcaster.com/
```

Application information response

An HTTP response is returned as follows:

- Header

```
HTTP/1.1 200 OK
Origin: http://cs.services.broadcaster.com/
```

- Body

```
<?xml version="1.0" encoding="UTF-8"?>
<service xmlns="urn:dial-multiscreen-org:schemas:dial"
         xmlns:hbbtv="urn:hbbtv:HbbTVCompanionScreen:2014" dialVer="1.7">
    <name>HbbTV</name>
    <options allowStop="false"/>
    <state>running</state>
    <additionalData>
        <hbbtv:X_HbbTV_App2AppURL>
            ws://192.168.1.11:992/hbbtv/84fa-9fd3-33a1-2481-9098-3ccd-de26-a223/
        </hbbtv:X_HbbTV_App2AppURL>
        <hbbtv:X_HbbTV_InterDevSyncURL>ws://192.168.1.11:991/css-cii
        </hbbtv:X_HbbTV_InterDevSyncURL >
        <hbbtv:X_HbbTV_UserAgent> Mozilla/5.0 (Linux armv7l) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/35.0.1916.153 Safari/537.36 OPR/22.0.1481.0 OMI/4.2.12.34.ALSAN3.16 HbbTV/1.4.1 (;
Sonic; VX600WDR; 1.14.0; ; dd5528b6-d0a9-40a8-acdb-21fa2eabeb2e; )
        </hbbtv:X_HbbTV_UserAgent>
    </additionalData>
</service>
```

14.8 Cross-Origin support

The HbbTV® terminal shall allow cross-origin requests to the HbbTV® UPnP device description (for the DIAL service) and the DIAL REST Service. It shall do this by implementing the resource processing model defined in the W3C Cross-Origin Resource Sharing recommendation [42] and authorising all HTTP requests made by a CS application or an HbbTV® application on another terminal to come from any origin. Specifically, when the HbbTV® terminal receives an HTTP request with request URL targetting the UPnP device description or the DIAL REST Service:

- If the request uses the OPTIONS method, the HbbTV® terminal shall process the request as a preflight request in accordance with clause 6.2 of the W3C Cross-Origin Resource Sharing recommendation [42] including the following HTTP headers in the HTTP response as appropriate: `Access-Control-Allow-Origin`, `Access-Control-Max-Age`, `Access-Control-Allow-Methods` and `Access-Control-Allow-Headers`. These headers shall be used to indicate that requests are permitted from any origin and to confirm that a CS application may use the HTTP POST.

- If the request contains an `Origin` header, then the HbbTV® terminal shall include an `Access-Control-Allow-Origin` header in the HTTP response. The value of this response header shall be either the asterisk character "`*`" or a case-sensitive match for the value of the `Origin` header from the HTTP request.

Annex A (normative): OIPF specification profile

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

Where constants are defined in in the OIPF DAE specification as input parameters and/or return values for methods or as values for properties, these constants shall be supported if any method or property is supported that uses them and if the constant is not explicitly excluded by name below. Although the constants defined in the OIPF DAE specification are expressed in JavaScript as properties, statements in Table A.1 that "Only the following properties shall be supported" do not apply to these constants.

Table A.1: Section-by-section profile of the OIPF DAE specification

Section, sub-section	Reference in DAE [1]	Status in HbbTV	Notes	Security
Gateway Discovery and Control	4.2	NI		
Application Definition				
Application definition	4.3 excluding sub-clauses	M(*)	Modified by the present document concerning the application boundary and access to privileged capabilities.	
Similarities between applications and traditional web pages	4.3.1	M		
Difference between applications and traditional web pages	4.3.2	NI	The present document defines a model supporting one application executing at one time and does not include background applications. See clause 6.1 of the present document.	
The application tree	4.3.3	NI		
The application display model	4.3.4	M(*)	The present document requires a different application visualization mode from those referred to here.	
The Security model	4.3.5	NI	See clause 11.1 of the present document.	
Inheritance of permissions	4.3.6	NI		
Privileged applications APIs	4.3.7	NI	Not applicable.	
Active applications list	4.3.8	NI	Not applicable.	
Widgets	4.3.9	NI		
Origin for Broadcast-delivered Documents	4.3.10	NI	See clause 6.3.2 of the present document.	
Resource Management				
Application lifecycle issues	4.4.1	NI	See clause 6.2.2.11 for terminal behaviour due to a lack of resources.	
Caching of application files	4.4.2	NI	See clause 6.1 of the present document concerning "background preloading" of applications.	
Memory usage	4.4.3	M	The `gc()` method is not included.	
Instantiating embedded object and claiming scarce system resources	4.4.4	M		
Media control	4.4.5	M(*)	Shall be modified as defined in clause A.2.1.	
Use of the display	4.4.6	M(*)	The present document defines a different application visualization mode than those in clause 4.4.6.	
Cross-application event handling	4.4.7	NI	Not applicable in the present document.	
Behaviour of the BACK key	4.4.7.1	M(*)	See clause A.2.6.4 of the present document.	
Tuner resources	4.4.8	M		

Section, sub-section	Reference in DAE [1]	Status in HbbTV	Notes	Security
Parental access control	4.5	M	- Approach A shall be supported for streaming on demand content. - Approach B shall be supported where CI Plus is supported. - Approach C shall always be supported. See clause 10.2.6.	
Content Download				
Download manager	4.6.1	M-D(*)	The `application/oipfStatusView` embedded object is not included.	Trusted
Content Access Download Descriptor	4.6.2	M-D		Trusted
Triggering a download	4.6.3	M-D		Trusted
Download protocol(s)	4.6.4	M-D		Trusted
Streaming CoD				
Unicast streaming	4.7.1	M(*)	All 3 methods listed in this clause shall be supported. An HTTP URL directly referencing the content to be streamed and an HTTP or HTTPS URL referencing a MPEG DASH MPD shall be supported as formats for a reference to unicast streaming content. The other formats listed in this clause are not required.	
HTTP Adaptive Streaming	4.7.1.1	NI	See clause 9.4 of the present document.	
Multicast streaming	4.7.1.2	NI		
Scheduled content				
Conveyance of channel list	4.8.1	M	Clause 4.8.1.2 is optional in DAE and not included in the present document.	Broadcast-related
Conveyance of channel list and list of scheduled recordings	4.8.2	M-P		Trusted
DLNA RUI Remote Control Function	4.9	NI		
Power Consumption	4.10	NI		
Display Model	4.11	M		
Application lifecycle				
Web applications	5.1.1.2	M	Web applications are equivalent to broadcast-independent applications in the present document.	
Applications started through an OITF-specific user interface	5.1.1.3	M		
Using the Application.createApplication API call	5.1.1.4	M	See clauses 6.2.2.6 and 9.2 of the present document.	
CE-HTML third party notifications	5.1.1.5	NI		
Starting applications from SD&S Signalling	5.1.1.6	NI		

Section, sub-section	Reference in DAE [1]	Status in HbbTV	Notes	Security
Applications started by the DRM agent	5.1.1.7	NI	Terminals should not start HbbTV® applications triggered by the DRM agent in order to avoid killing a currently running HbbTV® application which is trying to present the protected content. Instead it is recommend that applications trying to present protected content should handle DRM-specific UI themselves. Note that CI Plus application MMI (see clause 5.6.2 of the present document) has some conceptual similarities with this but uses a different presentation technology.	
Applications provided by the AG through the remote UI	5.1.1.8	NI		
Stopping an application	5.1.2	M		
Application Boundaries	5.1.3	NI	This subject is addressed in substantially more detail by clause 6.3 of the present document.	
Application announcement and signalling	5.2	NI		
Event Notification				
Event Notification Framework based on CEA 2014 - NotifSocket	5.3.1.1	NI		
Event Notification Framework based on CEA 2014 - XMLHttpRequest	5.3.1.1	M		None
Out of Session event notification	5.3.1.2	NI		
IMS Event Notification Framework	5.3.2	NI		
Formats				
Web Standards TV Profile	6.1	M	See clause A.2.5.7 Other modifications to the A/V Control object In clause 7.14.3.1, the definition of the property `onPlayPositionChanged( Integer position )` is changed as shown; The function that is called when change occurs in the play position of a channel due to the use of ~~trick play functions~~ random access. A.2.6 of the present document. NOTE: The reference to clause 3.2.5.1.7 of HTML5 is revised to be clause 3.2.4.1.7.	
Still Image Formats	6.2	M		
Media formats	6.3	M(*)	See clause 7 of the present document.	
SVG	6.4	NI		
APIs				

Section, sub-section	Reference in DAE [1]	Status in HbbTV	Notes	Security
Object Factory API	7.1	M(*)	Methods for creating objects not required by the present document are not included. The `requiredCapabilities` argument on the `createVideoBroadcastObject()` and `createVideoMpegObject()` methods shall not be used and can be ignored. Creation of embedded objects (both visual and non-visual) by using the `<object>` element in an HTML document or by using the DOM `createElement()` method and adding the resulting element to the application's DOM tree shall be supported. The `createMediaSynchroniser()` and `createCSManager()` methods defined in clause A.2.7 shall be supported. The extensions to `isObjectSupported()` defined in clause A.2.7 shall be supported.	None
Applications Management APIs				
The application/oipfApplicationManager embedded object	7.2.1	M(*)	The `getOwnerApplication`() method, `onLowMemory` and `onApplicationLoadError` properties (and corresponding DOM 2 events) shall be supported. All other properties, methods and DOM 2 events are not included.	None
The Application class	7.2.2	M(*)	The following properties and methods shall be supported: - `privateData` - `createApplication(URI,false)` - `destroyApplication()` - `show()` - `hide()` (broadcast independent applications should not call this method. Doing so may result in only the background being visible to the user). All other properties and methods are not included.	None
The ApplicationCollection class	7.2.3	NI		
The ApplicationPrivateData class	7.2.4	M(*)	The following properties and methods shall be supported: - `keyset` - `currentChannel` - `getFreeMem()` All other properties and methods are not included.	None

Section, sub-section	Reference in DAE [1]	Status in HbbTV	Notes	Security
The Keyset class	7.2.5	M(*)	The `otherKeys` and `maximumOtherKeys` properties and the `getKeyLabel` method are not included. The icons returned by the `getKeyIcon` method shall be 32 x 32 pixels.	None
New DOM events for application support	7.2.6	NI		None
Widget APIs	7.2.8	NI		
Configuration and Setting APIs				
The application/oipfConfiguration embedded object	7.3.1	M(*)	The `configuration` property shall be supported. All other properties, methods and events are not included.	None
The Configuration class	7.3.2	M(*)	Support for read-only access to the following properties is mandatory: - `preferredAudioLanguage` - `preferredSubtitleLanguage` - `preferredUILanguage` - `countryId` All other properties and methods are optional. The extensions to the `Configuration` class defined in clause A.2.20 shall be supported.	None
The LocalSystem class	7.3.3	NI		
The NetworkInterface class	7.3.4	NI		
The AVOutput class	7.3.5	NI		
The NetworkInterfaceCollection class	7.3.6	NI		
The AVOutputCollection class	7.3.7	NI		
The TunerCollection class	7.3.8	NI		
The Tuner class	7.3.9	NI		
The SignalInfo class	7.3.10	NI		
The LNBINfo class	7.3.11	NI		
The StartupInformation class	7.3.12	NI		
Content Download APIs				
application/oipfDownloadTrigger embedded object	7.4.1	M-D(*)	The definition of the `registerDownloadURL` method shall be modified as defined in clause A.2.19 of the present document. For the `registerDownload` and `registerDownloadURL` methods, the `downloadStart` parameter shall be ignored by terminals.	Trusted
Extensions to application/oipfDownloadTrigger	7.4.2	NI		

Section, sub-section	Reference in DAE [1]	Status in HbbTV	Notes	Security
application/oipfDownloadManager embedded object	7.4.3	M-D(*)	The `discInfo` property and the `updateRegisteredDownload`, `pause`, `resume` and `getDownloads` method arenot included. A download using FDP which has completed with errors shall be reported as successfully completed, in case `discard_file_on_error_flag` = 0 for this download (see clause H.4.2). The behaviour of the `reserve()` method is clarified by clause A.2.18 below.	Trusted
The Download class	7.4.4	M-D(*)	The `currentBitrate` property is not included. The `errorLevel` property shall be supported (see clause A.2.11 below). The `flaggedForDeletion` property defined in clause 8.2.4 shall be supported.	Trusted
The DownloadCollection class	7.4.5	M-D		Trusted
The DRMControlInformation class	7.4.6	M-D+ M-M	Mandatory if both Download and DRM features are supported - even if the supported DRM systems do not use the `<DRMControlInformation>` element inside the content access download descriptor. If the Download feature is supported and the terminal supports CI Plus and if the terminal is capable of providing downloaded content to the CICAM then these classes shall be supported - even if the CAS brought by a CICAM do not use the `<DRMControlInformation>` element inside the content access download descriptor.	
The DRMControlInfoCollection class	7.4.7	M-D+ M-M		
Content On Demand Metadata APIs	7.5	NI		
Content Service Protection API	7.6	M-C(*), M-M(*), M-P(*)	If the DRM feature is supported or if the terminal supports CI Plus then this is mandatory except as follows: - The `canRecordContent()` method is mandatory only if either or both of the preceding conditions apply and also the PVR feature is supported. - The `onDRMSystemStatusChange` property and the `DRMSystemStatus` method are not included. The `DRMSystemID` argument for the `sendDRMMessage()` method shall be specified and shall not be null	
Gateway Discovery and Control APIs	7.7	NI		
Communication Services APIs	7.8	NI		

Section, sub-section	Reference in DAE [1]	Status in HbbTV	Notes	Security
Parental access control APIs				
application/oipfParentalControlManager embedded object	7.9.1	M(*)	The `parentalRatingSchemes` property shall be supported. Other properties and methods are not included.	None
The ParentalRatingScheme class	7.9.2	M	A scheme supporting DVB-SI age based rating shall be supported. The `threshold.value` and `threshold.name` properties shall be `undefined` if the user has set no minimum age in the terminal's parental control system (i.e. the user will never be requested for their PIN) and the `threshold.scheme` property is `dvb-si`.	None
The ParentalRatingSchemeCollection class	7.9.3	M(*)	The `addParentalRatingScheme()` method is not included.	None
The ParentalRating class	7.9.4	M	For instances with a scheme of "`dvb-si`", the `name` property is a string containing an age in years, encoded as a decimal in the range "4" to "18" inclusive. For example, "13" means a programme that is rated suitable for persons of 13 years of age or older.	None
The ParentalRatingCollection class	7.9.5	M(*)	The `addParentalRating()` method shall be supported if the PVR feature is supported and is otherwise not included. All other features of the class shall be supported.	None
Scheduled Recording APIs				
application/oipfRecordingScheduler embedded object	7.10.1	M-P(*)	See clause A.2.24. Support for repeated recordings with the `recordAt()` method is not included and hence the `repeatDays` argument may be ignored.	Trusted
The ScheduledRecording class	7.10.2	M-P(*)	Only the following properties shall be supported: - `startPadding` - `endPadding` - `name` - `description` - `startTime` - `duration` - `state` - `parentalRatings` - `channel` - `programmeID` All other properties are not included.	Trusted
The ScheduledRecordingCollection class	7.10.3	M-P		Trusted
Extension to application/oipfRecordingScheduler for control of recordings	7.10.4	M-P(*)	The `recordings` property shall be supported and shall return recordings that are in-progress as well as ones that are scheduled or completed. Other properties, methods and events are not included.	Trusted

Section, sub-section	Reference in DAE [1]	Status in HbbTV	Notes	Security
The Recording class	7.10.5	M-P(*)	The following properties shall be supported: - `uri` - `id` - `recordingStartTime` - `recordingDuration` Since the `Recording` class implements the `ScheduledRecording` interface, the properties required to be supported from that interface as defined above are also required. All other properties are not included.	Trusted
The RecordingCollection class	7.10.6	NI		
The PVREvent class	7.10.7	NI		
The Bookmark class	7.10.8	NI		
The BookMarkCollection class	7.10.9	NI		
Remote Management APIs	7.11	NI		
Metadata APIs				
The application/oipfSearchManager embedded object	7.12.1	M(*)	The `guideDaysAvailable` and `onMetadataUpdate` properties are not included. For the `createSearch` method, only the value '1' of the `searchTarget` parameter is included.	Broadcast-related
The MetadataSearch class	7.12.2	M(*)	Only the value '1' of the `searchTarget` property is included. For the `createQuery` method, only the following case-insensitive values for the `field` parameter are included - "`Programme.startTime`", "`Programme.name`", "`Programme.programmeID`". These shall correspond to the properties of the same name. The `addRatingConstraint`, `addCurrentRatingConstraint` and `addChannelConstraint(ChannelList)` methods are not included. The `orderBy` method is not included - all search results shall be returned ordered first by channel, in the same order as presented to applications through a `ChannelList` object, then by start time in ascending order.	Broadcast-related
The Query class	7.12.3	M		Broadcast-related
The SearchResults class	7.12.4	M		Broadcast-related
The MetadataSearchEvent class	7.12.5	NI		
The MetadataUpdateEvent class	7.12.6	NI		
Scheduled content and hybrid tuner APIs				

Section, sub-section	Reference in DAE [1]	Status in HbbTV	Notes	Security
video/broadcast embedded object	7.13.1	M(*)	In the `setChannel()` method, the optional `contentAccessDescriptorURL` parameter may be ignored. The `setVolume`() and `getVolume`() methods and the `playerCapabilities` and `allocationMethod` properties are not included. The modifications in clause A.2.4 shall be supported.	See clause A.2.4
Extensions to video/broadcast for recording and timeshift	7.13.2	M(*), M-P(*)	OIPF DAE section 7.13.2 shall be replaced by the text in Annex A.2.4.7 and A.2.4.8. Terminals that support PVR shall support all of A.2.4.7 and A.2.4.8 Terminals that support time-shift of broadcast video shall support the following events and properties even if they do not support the full PVR option: - `onRecordingEvent` - `recordingState` - `playPosition` - `onPlayPositionChanged` - `playSpeed` - `onPlaySpeedChanged` - `playbackOffset` - `maxOffset` - `timeShiftMode` - `currentTimeShiftMode`	Broadcast-related
Extensions to video.broadcast for access to EIT p/f	7.13.3	M		Broadcast-related
Extensions to video/broadcast for playback of selected components	7.13.4	M	HbbTV terminals shall allow HbbTV applications to select media components in language(s) not supported by the terminal where there are no other reasons to refuse the selection (e.g. codec or subtitle character set not supported). For example, a terminal supporting French, German and Polish shall allow HbbTV applications to select media components in English, Italian or Chinese.	Broadcast-related
Extensions to video/broadcast for parental ratings errors	7.13.5	M		Broadcast-related
Extensions to video/broadcast for DRM rights errors	7.13.6	M-C	Mandatory if the terminal supports CI Plus.	
Extensions to video/broadcast for current channel information	7.13.7	M	Access to the `currentChannel` property by broadcast-independent applications shall return `null`.	Broadcast-related
Extensions to video/broadcast for creating Channel lists from SD&S fragments	7.13.8	NI		

Section, sub-section	Reference in DAE [1]	Status in HbbTV	Notes	Security
ChannelConfig class	7.13.9	M(*)	The `channelList` property shall be supported. Other properties, methods and events are not included.	Broadcast-related
ChannelList class	7.13.10	M(*)	The `getChannelBySourceID()` method is not included.	Broadcast-related
Channel class	7.13.11	M(*)	The following properties shall be supported: - `channelType` - `ccid` - `dsd` - `idType`- `nid` - `onid` - `tsid` - `sid` - `name` - `majorChannel` - `terminalChannel` All other properties and methods are not included. See clause 8.2.5 for more details regarding the properties `majorChannel` and `terminalChannel`.	Broadcast-related
Favourite lists	7.13.12, 7.13.13	NI		
Extensions to video/broadcast for channel scan	7.13.14	NI		
The ChannelScanEvent class	7.13.15	NI		
The ChannelScanOptions class	7.13.16	NI		
The ChannelScanParameters class	7.13.17	NI		
The DVBTChannelScanParameters class	7.13.18	NI		
The DVBSChannelScanParameters class	7.13.19	NI		
The DVBCChannelScanParameters class	7.13.20	NI		
Extensions to video/broadcast for synchronization	7.13.21	NI	Already included in clause 8.2.1 of the present document.	
The ATSCTChannelScanParameters class	7.13.22	NI		
Media Playback APIs				
The A/V Control object	7.14.1	M(*)	See clause A.2.5 of the present document The reference to the `<object>` element being defined in clause 4.8.4 of the HTML5 specification is revised to be clause 4.7.4.	None

Section, sub-section	Reference in DAE [1]	Status in HbbTV	Notes	Security
State diagram for A/V Control objects	7.14.1.1	M(*)	An `onPlaySpeedChanged` event shall be generated for all calls to the `play()` method regardless of the value returned by the method call and whether the play speed changes or not. In the present document, the `allocationMethod` property is not included but requirements for `DYNAMIC_ALLOCATION` shall apply. The transitions in the state diagram of figure 19 shall be mandatory.	None
Using an A/V Control object to play streaming content	7.14.1.2	M		None
Using an A/V Control object to play downloaded content	7.14.1.3	M(*)-D	Clarified by A.2.5.5 below.	Trusted
Using an A/V Control object to play recorded content	7.14.1.4	M-P		Trusted
Using the A/V Control object to play content fragments	7.14.1.5	M		None
User Input and the A/V Control object	7.14.1.6	M		
Extensions to A/V Control object for playback through Content-Access Streaming Descriptor	7.14.2	M		None
Extensions to A/V Control object for trickmodes	7.14.3	M(*)	Only the `onPlayPositionChanged` and `onPlaySpeedChanged` properties and events are required.	None
Extensions to A/V Control object for playback of selected components	7.14.4	M	HbbTV terminals shall allow HbbTV applications to select media components in language(s) not supported by the terminal where there are no other reasons to refuse the selection (e.g. codec or subtitle character set not supported). For example, a terminal supporting French, German and Polish shall allow HbbTV applications to select media components in English, Italian or Chinese.	None
Extensions to A/V Control object for parental rating errors	7.14.5	M	See clause 10.2.6.	None
Extensions to A/V Control object for DRM rights errors	7.14.6	M-M, M-C	Mandatory if the DRM feature is supported or if the terminal supports CI Plus. `onDRMRightsError` shall not cause a state transition of the A/V control object. It is the application's responsibility to stop the A/V Control object if that is the appropriate behaviour under the circumstances. If an error is generated because a suitable DRM system is not available then the `DRMSystemID` argument shall be undefined.	None

Section, sub-section	Reference in DAE [1]	Status in HbbTV	Notes	Security
Extensions to A/V Control object for playing media objects	7.14.7	M-D, M-P	Shall be supported if either the download or PVR features are supported. Calls to the `setSource()` method where id is a recording identifier shall result in the type attribute being set to "`video/mpeg`" regardless of the format in which the content is recorded.	Trusted
Extensions to A/V Control object for UI feedback of buffering A/V content	7.14.8	NI		
DOM events for A/V Control object	7.14.9	M		None
Playback of memory audio	7.14.10	M		None
Extensions to A/V Control object for media queuing	7.14.11	NI		
URI support and the queue method	7.14.11.1	NI		
Implementation Requirements on the Queue Method	7.14.11.2	NI		
Extensions to A/V Control object for volume control	7.14.12	NI		
Extensions to A/V Control object for resource management	7.14.13	NI		
Miscellaneous APIs				
application/oipfMDTF embedded object	7.15.1	NI		
application/oipfStatusView embedded object	7.15.2	NI		
application/oipfCapabilities embedded object	7.15.3	M	The `hasCapability()` method shall be supported with the profile names being the HbbTV® option strings as defined in clause 10.2.4. See clause A.2.1 for clarification of the behaviour of the `extraSDVideoDecodes` and `extraHDVideoDecodes` properties.	None
The Navigator class	7.15.4	M		None
Debug Print API	7.15.5	M		None
The StringCollection class	7.16.1.1	M		None
The IntegerCollection class	7.16.1.2	NI		
The Programme Class				

Section, sub-section	Reference in DAE [1]	Status in HbbTV	Notes	Security
Basics	7.16.2.1, 7.16.2.2	M(*)	The following properties are required: - `name` - `programmeID` - `programmeIDType` - `description` - `longDescription` - `startTime` - `duration` - `channelID` - `parentalRatings` All other properties and methods are not included. The constants defined in clause 7.16.2.1 shall be supported however support for CRIDs is outside the scope of the present document. The `count` parameter of the `findProgrammesFromStream` method of the `MetadataSearch` class is not included.	Broadcast-related
Metadata extensions to Programme	7.16.2.3	NI		
DVB-SI extensions to Programme	7.16.2.4	M		
Recording extensions to Programme	7.16.2.5	M-P		
The ProgrammeCollection class	7.16.3	M		Broadcast-related
The DiscInfo class	7.16.4	NI		
Extensions for playback of selected media components	7.16.5	M(*)	For mapping of the encoding property for MP4 FF content, the following additional sample track descriptions shall be included: "hvc1" → "video/mp4" "hev1" → "video/mp4" The `label` property defined in clause A.2.13 shall be supported. The `selectComponent()` and `unselectComponent()` methods shall be asynchronous. The `getComponents()` method shall always return fresh information. For example, in the case of an MPEG-2 transport stream, after a change to the PMT. The property defined in clause A.2.17 shall be supported.	

Section, sub-section	Reference in DAE [1]	Status in HbbTV	Notes	Security
Additional support for protected content	7.16.6	M-C, M-M	Mandatory if the DRM feature is supported or if the terminal supports CI Plus. NOTE: This clause defines extensions to both the `Recording` and `Download` classes. The extensions to the `Recording` class are mandatory if the PVR feature is supported. The extensions to the `Download` class are mandatory if the file download feature is supported.	Trusted
DLNA RUI Remote Control Function APIs	7.17	NI		
System integration aspects				
HTTP User-Agent header	8.1.1	NI	See clause 7.3.2.4.	
HTTP X-OITF-RCF-User-Agent header	8.1.2	NI		
Mapping from APIs to Protocols				
CoD Download Over HTTP	8.2.1	M-D		
CoD Unicast Streaming with SIP Session Management	8.2.2	NI		
Scheduled Content Multicast Streaming with SIP Session Management	8.2.3	NI		
Communication Services with SIP Session Management	8.2.4	NI		
CoD Unicast Streaming over RTP and HTTP	8.2.5	M(*)		
General	8.2.5.1	M(*)	Only for the HTTP protocol	
CoD Media Queuing	8.2.5.2	M		
Scheduled content Multicast Streaming	8.2.6	NI		
URI Schemes and their usage	8.3	M	The `http:`, `https:` and `dvb:` URL schemes shall be supported as defined in this clause.	
Media Fragments Support	8.3.1	M		
Mapping from APIs to Content Formats				
Character Conversion	8.4.1	M		
AVComponent	8.4.2	M(*)	Only for properties that are required by the present document. Statements that a property "may" be derived in a particular way shall be read as "shall" be derived in that way For `AVComponent`s corresponding to an MPEG DASH Adaptation Set, the `language` property shall be what is encoded in the MPD which may be an ISO 639-1 [60] 2-character language code and not an ISO 639-2 [61] 3-character language code.	

Section, sub-section	Reference in DAE [1]	Status in HbbTV	Notes	Security
Channel	8.4.3	M(*)	Only the requirements about channels of type `ID_DVB_*` applies and only then for properties that are required by the present document.	
Programme, ScheduledRecording, Recording and Download	8.4.4	M(*)	Only for properties that are required by the present document.	
Exposing Audio Description Streams as AVComponent objects	8.4.5	M(*)	This only applies to the extent that the terminal supports audio description.	
HTML5 Media Element Mapping	8.4.6	M(*)	See clause A.2.12 of the present document.	
DLNA RUI Remote Control Function implementation	8.5	NI		
Capabilities				
Minimum DAE capability requirements	9.1	NI	See clause 10.2.1 in the present document.	
SSL/TLS Requirements	9.1.1	NI	9.1.1 is replaced by clause 11.2 of the present document.	
Default UI profiles	9.2	M(*)	Clause 10.2.4 of the present document requires support for `OITF_HD_UIPROF` defined in this clause. That in turn requires support for `OITF_SDEU_UIPROF`. The definition of that is modified as follows, `<security protocolNames="ssl tls">true</security>` is replaced by `<security protocolNames="tls">true</security>`	
CEA-2014 capability negotiation and extensions				
Tuner/broadcast capability indication	9.3.1	M		
Broadcasted content over IP capability indication	9.3.2	NI		
PVR capability indication	9.3.3	M-P		
Download Cod capability indication	9.3.4	M-D		
Parental ratings	9.3.5	M		
Extended A/V API support	9.3.6	M		
OITF Metadata API support	9.3.7	M		
OITF Configuration API support	9.3.8	M		
Communication Services API Support	9.3.9	NI		
DRM capability indication	9.3.10	M		
Media profile capability indication	9.3.11	M(*)	Valid values for the "`name`" attribute of the `<video_profile>` element shall include ones with an underscore and the subtitle format name appended to the end of what is defined in this clause. Subtitle format names are defined in clause 7.3.1.5 of the present document.	
Remote diagnostics support	9.3.12	NI		
SVG	9.3.13	NI		
Third party notification support	9.3.14	NI		
Multicast Delivery Terminating Function support	9.3.15	NI		
Other capability extensions	9.3.16	M		
HTML5 video	9.3.17	M		
DLNA RUI Remote Control Function support	9.3.18	NI		
Power Consumption	9.3.19	NI		
Widgets	9.3.20	NI		
Buffer control of AV content playback API support	9.3.21	NI		

Section, sub-section	Reference in DAE [1]	Status in HbbTV	Notes	Security
Temporal Clipping	9.3.22	M		
Capability Elements from other schemas	9.3.23	M		
Pointer support	9.3.24	M	See clause 10.2.2.2.	
Security				
OITF requirements	10.1.1	NI		
Server requirements	10.1.2	NI		
Specific security requirements for privileged JavaScript APIs	10.1.3	NI		
Permission names	10.1.4	NI		
Loading documents from different domains	10.1.5	M		
User Authentication	10.2	M(*)	HTTP Basic and Digest Authentication as defined in clause 5.4.1 of the OIPF CSP specification [5] shall be supported. Other forms of user authentication from clause 5 of the OIPF CSP specification are not included.	
DLNA RUI Remote Control	10.3	NI		
DAE Widgets	11	NI		
Performance				
Graphics Performance	12.1	M(*)	See clause A.2.16.	
Content Access Descriptor Syntax and Semantics				
Content Access Download Descriptor Format	E.1	M-D	Required with the extensions defined in clause A.2.25	
Content Access Streaming Descriptor Format	E.2	M(*)	Required with the extensions defined in clause A.2.25	
Abstract Content Access Descriptor Format	E.3	M	Required as the base descriptor for E.1 and E.2. When parsing a `<ParentalRating>` element, the content of that element is used as the name property of the JavaScript `ParentalRating` object. Valid values are the ones defined as valid for a `ParentalRating` object using the indicated scheme. The URL in the `<ContentURL>` element shall be an absolute URL. Relative URLs shall not be used in this element.	
Capability Extensions Schema	F	M		
Client Channel Listing Format	G	NI		
Display Model	H	M(*)	Modified by clause A.2.14 concerning scaling and clipping of video when not in full screen mode.	
Backwards Compatible Profile Of HTML5 Media Elements	I	NI		
DLNA RUI Remote Control Function Sequences	J	NI		
Collections	K	M		
SVG Video Tag Support	L	NI		
Changes to section 5.6.2 of CEA-2014-A	O	NI		

Table A.2: Key to security column

Security	Description
none	All applications shall have access to the referenced API.
trusted	Only trusted applications as defined in clause 11.1 shall have access to the referenced API. If other applications or web pages try to use this API, the terminal shall throw an error with the `name` property set to `SecurityError` (see clause 10.1.1 of the OIPF DAE specification [1]). Note that for embedded objects, untrusted applications may acquire instances of them without restrictions, either through the object factory or by using HTMLObjectElements. Security restrictions are enforced only when the application attempts to access properties or execute functions on the objects.
broadcast-related	Broadcast-related applications shall have access to the referenced API regardless of whether they are trusted or not. If other applications or web pages try to use this API, the terminal shall throw an error with the `name` property set to `SecurityError` (see clause 10.1.1 of the OIPF DAE specification [1]). Note that for embedded objects, untrusted broadcast-independent applications may acquire instances of them without restrictions, either through the object factory or by using `HTMLObjectElement`s. Security restrictions are enforced only when the application attempts to access properties or execute functions on the objects.
n/a (for optional APIs)	The security level for optional APIs is the manufacturer's decision. If such APIs are provided, they should have at least a security level of "trusted". Further restrictions may be added.

Table A.3: Key to status column

Status	Meaning
M	Mandatory.
M-C	Mandatory if CI Plus is supported for protected content via broadcast. Support of the related section/sub-section in Table A.1 is not expected if CI Plus support is not indicated according to clause 10.2.4.
M-D	Mandatory if the download feature supported otherwise not included.
M-M	Mandatory if the DRM feature is supported otherwise not included. Support of the related section/sub-section in Table A.1 is not expected if the support of the DRM feature is not indicated according to clause 10.2.4. See note 2.
M-P	Mandatory if the PVR feature is supported otherwise not included.
NI	Not included.
NOTE 1: Any of the above may be post-fixed with (*) where only some parts of the section or sub-section are required in the present document. NOTE 2: A device supporting CI Plus is not expected to support all the APIs required for the DRM feature.	

A.2 Modifications, extensions and clarifications to volume 5

A.2.1 Resource management

In clause 4.4.5 of the OIPF DAE specification [1], the `STATIC_ALLOCATION` model is not included in the present document. All resource allocation is under the `DYNAMIC_ALLOCATION` model.

NOTE 1: The policy for managing hardware resources defined here that applies to the A/V Control object and video/broadcast objects (first-come, first-served) is intentionally the exact opposite of the policy defined for the HTML 5 media element in clause 9.6.2 of the present document.

If the resources that would be needed by an HTML5 media element to present media are in use by either i) an A/V Control object or ii) a video/broadcast object or iii) the terminal for presenting broadcast content not under the control of a video/broadcast object, and the media element requiring the resource and the current media element owning the resource have not been added to the same media synchroniser object, then the request to present media through the media element shall fail with a `MediaError` with code `MEDIA_ERR_DECODE`.

If the resources that would be needed by an A/V Control object or a video/broadcast object are in use by an HTML5 media element, and the media element requiring the resource and the current media element owning the resource have not been added to the same media synchroniser object, then the request to present media through the object shall fail. For an A/V control object, the object shall go to playState 6 with the error property being 3, "insufficient resources".

For a video/broadcast object, this shall be reported by an `onChannelChangeError` with `errorState` 11, "insufficient resources are available to present the given channel (e.g. a lack of available codec resources)".

NOTE 2: Component selection behaviour is different after a media element has been added to a `MediaSynchroniser` object. This behaviour ensures that only one audio or video component is decoded simultaneously. See clause 10.2.7.3 for more details.

The properties `extraSDVideoDecodes` and `extraHDVideoDecodes` shall return the number of additional A/V decoders available at the time the application reads the properties and that can be used with either the A/V Control object or HTML5 media elements to render additional broadband streams.

A.2.2 Void

A.2.3 Void

A.2.4 Extensions to the video/broadcast object

A.2.4.1 State machine and related changes

This clause describes a set of changes to the state machine and related text for the video/broadcast object defined in clause 7.13.1.1 of the OIPF DAE specification [1].

- Calling the `setChannel()` method from any state of the video/broadcast object with a null argument shall cause the application to transition to a broadcast-independent application (as described in clause 6.2.2.6). This is in addition to what is required by OIPF - e.g. causing the video/broadcast object to transition to the unrealized state and releasing any resources used for decoding video and/or audio. Hence the `setChannel(null)` and `release()` methods do not have the same behaviour in the present document.

- A video/broadcast object with a CSS rule of `display:none` shall not be loaded and hence shall not be decoding audio or video.

- In table 12, “State transitions for the video/broadcast embedded object”, the following row is modified as shown underlined;

Old State	Trigger	New State	State Transition Events	Description
Stopped	bindToCurrentChannel()	Connecting	PlayStateChange	~~Video and audio presentation is enabled~~ The terminal starts to present the current channel.

- In clause 7.13.1.3 of the OIPF DAE specification [1], the definition of the `bindToCurrentChannel()` method is modified as shown:

  If the video/broadcast object is in the unrealized state and ~~video from~~ exactly one channel is currently being presented by the OITF then this binds the video/broadcast object to that ~~video~~channel (even if the current channel does not contain video and/or audio). If more than one channel is currently being presented by the OITF then this binds the video/broadcast object to the channel whose audio is being presented.

  If the video/broadcast object is in the stopped state then this restarts presentation of ~~video and audio from~~ the current channel under the control of the video/broadcast object. ~~If video from more than one channel is currently being presented by the OITF then this binds the video/broadcast object to the channel whose audio is being presented.~~

A.2.4.2 Access to the video/broadcast object

The following rules and clarifications shall apply to the video/broadcast object.

Broadcast-related applications shall have full access to the video/broadcast object. If a new broadcast service is selected then this may result in the broadcast-related application being killed as defined in clause 6.2.2.2. As defined in

clause 6.2.2.2, selecting MPEG programs which are not broadcast services and which do not contain an AIT will not cause the running broadcast-related application to be killed.

Broadcast-independent applications shall be able to use the video/broadcast object as follows.

- The following properties and methods shall have no restrictions: `createChannelObject()`, `onChannelChangeSucceeded`, `onChannelChangeError`, `onPlayStateChange`, `addEventListener()`, `removeEventListener()`, `width` and `height`.
- The `setChannel()` method shall trigger the behaviours defined in clause 6.2. If the method is used to select a broadcast service then this may result in the application becoming a broadcast-related application. If the `setChannel()` method is used to access an MPEG program which is not a broadcast service and which does not contain an AIT, then there are no restrictions and no consequences for the application lifecycle.
- The following methods shall always throw a "Security Error" (as defined in clause 10.1.1 of the OIPF DAE specification [1]): `getChannelConfig()`, `bindToCurrentChannel()`, `prevChannel()` and `nextChannel()`.
- The following methods shall have no effect: `setFullScreen()`, `release()`, and `stop()`.
- The object shall always be in the unrealized or connecting states unless connected to an MPEG program which is not a broadcast service and which does not contain an AIT.

Terminals shall only support one active instance of a video/broadcast object at any time. "Active" means here that the video/broadcast object is either in the `connecting` or the `presenting` state. Trying to activate an instance of a video/broadcast object (through a call to `bindToCurrentChannel()` or a `setChannel()` call) while another instance is already active shall fail and result in an error returned to the application through a `ChannelChangeError` event.

A.2.4.3 Support for quiet service selection

A.2.3.4.1 Quiet service selection

The following changes shall apply to the video/broadcast object to support “quiet” service selection.

```
void setChannel( Channel channel, Boolean trickplay, String contentAccessDescriptorURL,
                 Number quiet )
```

Description	Requests the OITF to switch a (logical or physical) tuner to the channel specified by channel and render the received broadcast content in the area of the browser allocated for the `video/broadcast` object. If the channel specifies an `idType` attribute value which is not supported by the OITF or a combination of properties that does not identify a valid channel, the request to switch channel SHALL fail and the OITF SHALL trigger the function specified by the `onChannelChangeError` property, specifying the value 0 ("Channel not supported by tuner") for the errorState, and dispatch the corresponding DOM event (see below). If the channel specifies an `idType` attribute value supported by the OITF, and the combination of properties defines a valid channel, the OITF SHALL relay the channel switch request to a local physical tuner that is currently not in use by another `video/broadcast` object and that can tune to the specified channel. If no tuner satisfying these requirements is available (i.e. all physical tuners that could receive the specified channel are in use), the request SHALL fail and OITF SHALL trigger the function specified by the `onChannelChangeError` property, specifying the value '2' ("tuner locked by other object") for the `errorState` and dispatch the corresponding DOM event (see below). If multiple tuners satisfying these requirements are available, the OITF selects one. If the channel specifies an IP broadcast channel, and the OITF supports idType ID_IPTV_SDS or ID_IPTV_URI, the OITF SHALL relay the channel switch request to a logical 'tuner' that can resolve the URI of the referenced IP broadcast channel. If no logical tuner can resolve the URI of the referenced IP broadcast channel, the request SHALL fail and the OITF SHOULD trigger the function specified by the `onChannelChangeError` property, specifying the value 8 ("cannot resolve URI of referenced IP channel") for the errorState, and dispatch the corresponding DOM event. The optional attribute `contentAccessDescriptorURL` allows for the inclusion of a Content Access Streaming Descriptor (the format of which is defined in Annex E.2) to provide additional information for dealing with IPTV broadcasts that are (partially) DRM-protected. The descriptor may for example include Marlin action tokens or a previewLicense. The attribute SHALL be `undefined` or `null` if it is not applicable. If the attribute `contentAccessDescriptorURL` is present, the trickplay attribute shall take a value of either `true` or `false`. If the Transport Stream cannot be found, either via the DSD or the (ONID,TSID) pair, then a call to `onChannelChangeError` with `errorstate`=5 ("unknown channel") SHALL be triggered, and the corresponding DOM event dispatched. If the OITF succeeds in tuning to a valid transport stream but this transport stream does not contain the requested service in the PAT, the OITF SHALL remain tuned to that location and SHALL trigger a call to `onChannelChangeError` with `errorstate`=12 ("specified channel not found in transport stream"), and dispatch the corresponding DOM event. If, following this procedure, the OITF selects a tuner that was not already being used to display video inside the `video/broadcast` object, the OITF SHALL claim the selected tuner and the associated resources (e.g., decoding and rendering resources) on behalf of the video/broadcast object. If all of the following are true: • the video/broadcast object is successfully switched to the new channel • the channel is a locally defined channel (created using the `createChannelObject` method) • the new channel has the same tuning parameters as a channel already in the channel list in the OITF • the `idType` is a value other than `ID_IPTV_URI` then the result of this operation SHALL be the same as calling `setChannel` with the `channel` argument being the corresponding channel object in the channel list, such that: • the values of the properties of the `video/broadcast` object `currentChannel` SHALL be the same as those of the channel in the channel list • any subsequent call to `nextChannel` or `prevChannel` SHALL switch the tuner to the next or previous channel in the favourite list or channel list as appropriate, as described in the definitions of these methods

	Otherwise, if any of the above conditions is not true, then: • the values of the properties of the `video/broadcast` object `currentChannel` SHALL be the same as those provided in the `channel` argument to this method, updated as defined in section 8.4.3 • the channel is not considered to be part of the channel list The resulting current channel after any subsequent call to `nextChannel()` or `prevChannel()` is implementation dependent, however all appropriate functions SHALL be called and DOM events dispatched.The OITF SHALL visualize the video content received over the tuner in the area of the browser allocated for the `video/broadcast` object. If the OITF cannot visualize the video content following a successful tuner switch (e.g., because the channel is under parental lock), the OITF SHALL trigger the function specified by the `onChannelChangeError` property with the appropriate `channel` and `errorState` value, and dispatch a corresponding DOM event (see below). If successful, the OITF SHALL trigger the function specified by the `onChannelChangeSucceeded` property with the given channel value, and also dispatch a corresponding DOM event.	
Arguments	`channel`	The channel to which a switched is requested. If the channel object specifies a ccid, the ccid identifies the channel to be set. If the channel does not specify a ccid, the idType determines which properties of the channel are used to define the channel to be set, for example, if the channel is of type `ID_IPTV_SDS` or `ID_IPTV_URI`, the `ipBroadcastID` identifies the channel to be set. If null, the `video/broadcast` object SHALL transition to the unrealized state and release any resources used for decoding video and/or audio. A `ChannelChangeSucceeded` event SHALL be generated when the operation has completed.
	`trickplay`	Optional flag indicating whether resources SHOULD be allocated to support trick play. This argument provides a hint to the receiver in order that it may allocate appropriate resources. Failure to allocate appropriate resources, due to a resource conflict, a lack of trickplay support, or due to the OITF ignoring this hint, SHALL have no effect on the success or failure of this method. If trickplay is not supported, this SHALL be indicated through the failure of later calls to methods invoking trickplay functionality. The `timeShiftMode` property defined in section 7.13.2.2 shall provide information as to type of trickplay resources that should be allocated. If argument `contentAccessDescriptorURL` is included then the `trickplay` argument SHALL be included.
	`contentAccessDescriptorURL`	Optional argument containing a Content Access Streaming descriptor (the format of which is defined in Annex E.2) that can be included to provide additional information for dealing with IPTV broadcasts that are (partially) DRM-protected. The argument SHALL be undefined or null if it is not applicable.
	`quiet`	Optional flag indicating whether the channel change operation shall be carried out quietly, as described in clause A.2.4.3.2 below. Valid values are: • 0: Normal channel change • 1: Normal channel change with no UI displayed • 2: Quiet channel change All other values shall cause a normal channel change to occur.

A.2.4.3.2 Changes to the video/broadcast object

For the `setChannel()` method, if the `quiet` argument is set to 0 or is omitted then the terminal shall execute the channel change operation normally. Typically, this means that the viewer experience is exactly as if they had initiated a standard channel change operation using any of the terminal's inherent channel navigation mechanisms, e.g. using the Ch+ or Ch- keys or numeric entry. This may be reflected in (but not restricted to):

- Presentation of any channel information on the terminal's front panel.
- Presentation of any now/next information or channel banner.
- The channel relative to which any navigation, such as Ch+/- or calls to `prevChannel()` or `nextChannel()`, is performed.

If the `quiet` argument is set to 1 then the terminal shall execute the channel change operation but shall not present any channel banner that is usually displayed by the terminal.

If the `quiet` argument is set to 2 then the terminal shall execute the channel change operation quietly. This means that the terminal shall suppress the presentation of all information usually presented during a channel change operation. In addition, the channel selected by the last normal channel change operation shall be used for all relevant interaction with the terminal by the viewer, which may include (but is not restricted to):

- Presentation of any channel information on the terminal's front panel.
- Presentation of any now/next information or channel banner.
- The channel relative to which any navigation, such as Ch+/- or calls to `prevChannel()` or `nextChannel()`, is performed.

The channel which is reported to HbbTV applications as the current channel shall be the actual channel currently selected, regardless of the type of channel change by which it was selected.

After a channel change where the `quiet` argument is set to a value of 1 or 2, the application lifecycle shall be identical to a normal channel change, i.e. one where the `quiet` argument is set to the value 0 or omitted. The lifecycle of the application shall follow the rules defined in sections 6.2.2.2 and 6.2.2.3 of the present document. The AIT of the new channel shall be obeyed following the channel change operation.

A.2.4.4 Definition of "delivery system descriptor"

The definitions of the `createChannelObject(Integer idType, String dsd, Integer sid)` method on the video/broadcast object, and the `dsd` attribute on the returned `Channel` object, both refer to the "delivery system descriptor". This "delivery system descriptor" shall be as follows:

For a DVB-T channel, the "delivery system descriptor" shall be a terrestrial_delivery_system_descriptor.

For a DVB-T2 channel, the "delivery system descriptor" shall be a T2_delivery_system_descriptor which shall include at least one centre_frequency field.

For a DVB-S channel, the "delivery system descriptor" shall be a satellite_delivery_system_descriptor.

For a DVB-S2 channel the "delivery system descriptor" shall be either a satellite_delivery_system_descriptor, or the concatenation of an S2_satellite_delivery_system_descriptor and a satellite_delivery_system_descriptor, in that order.

For a DVB-S2X channel, the "delivery system descriptor" shall be an S2X_satellite_delivery_system_descriptor.

For a DVB-C channel, the "delivery system descriptor" shall be a cable_delivery_system_descriptor.

For a DVB-C2 channel that does not use channel bundling, the "delivery system descriptor" shall be a C2_delivery_system_descriptor.

For a DVB-C2 channel that uses channel bundling, the "delivery system descriptor" shall be the concatenation of one or more C2_bundle_delivery_system_descriptor.

The descriptors referred to above are all defined in clause 6.2.13 and clause 6.4.5 of EN 300 468 [16].

A.2.4.5 Other modifications to the video/broadcast object

In clause 7.13.2.2, the definition of the property `onPlayPositionChanged( Integer position )` is changed as shown;

> The function that is called when change occurs in the play position of a channel due to the use of ~~trick play functions~~ random access functions.

A.2.4.6 Support for creating audio and video components

The following changes shall apply to the video/broadcast object to support creating `AVAudioComponent` and `AVVideoComponent` objects.

NOTE: Delivering A/V streams signaled as private data and referencing them via these methods is intended to be used for cases when those A/V streams are only appropriate for presentation under the control of an HbbTV application.

`AVAudioComponent createAVAudioComponent( Integer componentTag, String language, Boolean audioDescription, Integer channels, String encoding)`		
Description	The method creates an `AVAudioComponent` object for an elementary stream in the currently selected broadcast service that is not recognisable from the signalling as being an `AVVideoComponent`, an `AVAudioComponent` or an `AVSubtitleComponent`.(See clause 8.4.2 of the OIPF DAE specification for more information). Successfully created objects shall be usable identically to `AVAudioComponent` objects returned when the `getComponents()` method is called with the argument `COMPONENT_TYPE_AUDIO`. Null shall be returned if either no component with the specified value of `componentTag` is present in the currently selected broadcast service or if the value of `componentTag` specifies a component that already has an `AVVideoComponent`, an `AVAudioComponent` or an `AVSubtitleComponent`.	
Arguments	`componentTag`	A component tag identifying the elementary stream for which the `AVAudioComponent` object is to be created. This shall be matched against a component tag in a stream identifier descriptor in the ES loop of a stream in the PMT.
	`language`	The value to be returned from the `language` property of the object that is created. This should be an ISO 639-2 language code.
	`audioDescription`	The value to be returned from the `audioDescription` property of the object that is created.
	`channels`	The value to be returned from the `channels` property of the object that is created. The value indicates the number of main channels present in the stream (e.g. 5 for 5.1) excluding any potential low frequency effects channels.
	`encoding`	The value to be returned from the `encoding` property of the object that is created.

`AVVideoComponent createAVVideoComponent( Integer componentTag, Number aspectratio, String encoding)`

Description	The method creates an `AVVideoComponent` object for an elementary stream in the currently selected broadcast service that is not recognisable from the signalling as being an `AVVideoComponent`, an `AVAudioComponent` or an `AVSubtitleComponent`.(See clause 8.4.2 of the OIPF DAE specification for more information). Successfully created objects shall be usable identically to `AVVideoComponent` objects returned when the `getComponents()` method is called with the argument `COMPONENT_TYPE_VIDEO`. Null shall be returned if either no component with the specified value of `componentTag` is present in the currently selected broadcast service or if the value of `componentTag` specifies a component that already has an `AVVideoComponent`, an `AVAudioComponent` or an `AVSubtitleComponent`.	
Arguments	`componentTag`	A component tag identifying the elementary stream for which the `AVVideoComponent` object is to be created. This shall be matched against a component tag in a stream identifier descriptor in the ES loop of a stream in the PMT.
	`aspectratio`	The value to be returned from the `aspectratio` property of the object that is created.
	`encoding`	The value to be returned from the `encoding` property of the object that is created.

A.2.4.7 Extensions to video/broadcast for time-shift

A.2.4.7.1 General

If a terminal has indicated support in its capability description for recording functionality (i.e. by giving value `true` to the `<recording>` element as specified in OIPF DAE [1] section 9.3.3), the terminal shall support the following additional constants, properties and methods on the video/broadcast object, in order to start a time-shift of the current broadcast.

Note that this functionality is subject to the security model as specified in OIPF DAE [1] section 10.1.

Terminals may restrict access to the time-shift methods to those applications that are signalled as safe to run when time-shifting, i.e. those signaled in the AIT with an application_recording_descriptor and both the trick_mode_aware_flag and the time_shift_flag set to '1' as described in clause 7.2.3.1.

The properties and methods defined in this clause are used when the content presented in the video/broadcast object is being time-shifted.

A.2.4.7.2 Constants

Name	Value	Use
`POSITION_START`	0	Indicates a playback position relative to the start of the buffered content.
`POSITION_CURRENT`	1	Indicates a playback position relative to the current playback position.
`POSITION_END`	2	Indicates a playback position relative to the end of the buffered content (co-incident with the live playback position).

A.2.4.7.3 Properties

```
function onPlaySpeedChanged( Number speed )
```

The function that is called when the playback speed of a channel changes during timeshift.

The specified function is called with one argument, speed, which is defined as follows:

- Number speed – the playback speed of the media at the time the event was dispatched.

```
ScheduledRecordingCollection getInProgressRecordings()
```

The function that is called when a change occurs in the play position of a channel due to the use of trick play functions during timeshift.

The specified function is called with one argument, position, which is defined as follows:

- Integer position – the playback position of the media at the time the event was dispatched, measured from the start of the timeshift buffer in milliseconds. If the play position cannot be determined, this argument takes the value `undefined`.

`ScheduledRecordingCollection getInProgressRecordings()`

The function that is called when a change occurs in the play position of a channel due to the use of trick play functions during timeshift.

The specified function is called with one argument, position, which is defined as follows:

- Integer position – the playback position of the media at the time the event was dispatched, measured from the start of the timeshift buffer in milliseconds. If the play position cannot be determined, this argument takes the value `undefined`.

`readonly Integer playbackOffset`

Returns the playback position during timeshift, specified as the number of seconds between the live broadcast and the currently rendered position in the timeshift buffer, where a value of zero means that the broadcast is not being timeshifted or is playing from the live point in a timeshift buffer.

When the `currentTimeShiftMode` property has the value 0, the value of this property is `undefined`.

`readonly Integer maxOffset`

Returns the maximum playback offset, in seconds of the live broadcast, which is supported for the currently rendered broadcast. If the maximum offset is unknown, the value of this property shall be undefined.

NOTE: This value gives the size of the timeshift buffer.

When the `currentTimeShiftMode` property has the value 0, the value of this property is `undefined`.

`readonly Integer playPosition`

If the value of the `currentTimeShiftMode` property is 1, the current playback position of the media, measured in milliseconds from the start of the timeshift buffer.

`readonly Integer playSpeed`

The current play speed of the media.

`readonly Number playSpeeds[]`

Returns the ordered list of playback speeds, expressed as values relative to the normal playback speed (1.0), at which the currently specified content can be played (as a time-shifted broadcast in the video/broadcast object), or undefined if the supported playback speeds are not known.

If timeshift is supported by the terminal, the `playSpeeds` array shall always include at least the values 1.0 and 0.0.

This property may include the playback speeds that this broadcast content could be played back after being recorded, but only if they also apply to playback of the content when timeshifted.

`function onplaySpeedsArrayChanged()`

The function that is called when the `playSpeeds` array values have changed. An application that makes use of the `playSpeeds` array needs to read the values of the `playSpeeds` property again.

`Integer timeShiftMode`
The time shift mode indicates the mode of operation for support of timeshift playback in the video/broadcast object. Valid values are: **Value** | **Description** 0 | Timeshift is turned off. 1 | Timeshift shall use “local resource”. If the property is not set, the default value of the property is 1.

`readonly Integer currentTimeShiftMode`
When timeshift is in operation the property indicates which resources are currently being used. Valid values are: **Value** | **Description** 0 | Timeshift is turned off. 1 | Timeshift shall use “local resource”.

A.2.4.7.4 Methods

`Boolean pause()`	
Description	Pause playback of the broadcast. This is equivalent to `setSpeed(0)`.

`Boolean resume()`	
Description	Resumes playback of the time-shifted broadcast. This is equivalent to `setSpeed(1)`.

`Boolean setSpeed( Number speed )`

Description	Sets the playback speed of the time-shifted broadcast to the value speed. If the value of the `timeShiftMode` property is 0 or if trick play is not supported for the channel currently being rendered, this method shall return `false` and have no effect. If `speed` is a value less than 1.0 and the broadcast was not previously being time-shifted, this method shall start recording the broadcast that is currently being rendered live (i.e. not time-shifted) in the video/broadcast object. If the terminal has buffered the 'live' broadcasted content, the recording starts with the content that is currently being rendering in the video/broadcast object. Acquiring the necessary resources to start recording the broadcast may be an asynchronous operation, and presentation of the broadcast may not be affected until after this method returns; applications may receive updates by registering a listener for `PlaySpeedChanged` events as defined in A.2.26.5. If speed is a value greater than 1.0 and the broadcast was not previously being time-shifted, this method shall have no effect and shall return `false`. When playback is paused (i.e. by setting the play speed to 0), the last decoded video frame shall be shown. If the time-shifted broadcast cannot be played at the desired speed, specified as a value relative to the normal playback speed, the playback speed will be set to the best approximation of `speed`. If there is no change to the play speed as a result of the method call, it shall return `false`. Unless specified otherwise above, this method shall return `true`. After initial operation of `setSpeed()` several events may affect the content playback. If during fast forward the end of stream is reached the playback shall resume at normal speed and a `PlaySpeedChanged` event generated. If the end of the timeshift buffer is reached due to end of content the playback shall automatically be paused and a `PlaySpeedChanged` event generated. Any resources used for time-shifting shall not be discarded. If during rewinding the playback reaches the point that it cannot be rewound further, playback shall resume at normal speedand a `PlaySpeedChanged` event generated. A `PlaySpeedChanged` event shall be generated when the operation has completed, regardless of the success of the operation. If the operation fails, the argument of the event shall be set to the previous play speed.	
Arguments	*speed*	The desired relative playback speed, specified as a float value relative to the normal playback speed of 1.0. A negative value indicates reverse playback.

```
Boolean seek( Integer offset, Integer reference )
```

Description	Sets the playback position of the time-shifted broadcast that is being rendered in the video/broadcast object to the position specified by the offset and the reference point as specified by one of the constants defined in A.2.26.2. Playback of live content is resumed if the new position equals the end of the time-shift buffer. Returns `true` if the playback position is a valid position to seek to, `false` otherwise. If time-shift is not supported for the current channel (e.g. due to restrictions imposed by a conditional access or DRM system) or the broadcast is not currently being time-shifted or if the position falls outside the time-shift buffer, the terminal shall ignore the request to seek and shall return the value `false`. Applications are not required to pause playback of the broadcast or take any other action before calling `seek()`. This operation may be asynchronous, and presentation of the video may not be affected until after this method returns. For this reason, a `PlayPositionChanged` event shall be generated when the operation has completed, regardless of the success of the operation. If the operation fails, the argument of the event shall be set to the previous play position. After initial operation of `seek()` several events may affect the content playback. If during this operation the live playback position is reached the playback shall resume at normal speed and a `PlaySpeedChanged` event generated. If the timeshift buffer cannot be rewound any further, the playback shall automatically be paused and a `PlaySpeedChanged` event generated. Any resources used for time-shifting shall not be discarded.	
Arguments	*offset*	The offset from the reference position, in seconds. This can be either a positive value to indicate a time later than the reference position or a negative value to indicate time earlier than the reference position.
	reference	The reference point from which the offset shall be measured. The reference point can be either `POSITION_CURRENT`, `POSITION_START`, or `POSITION_END`.

`Boolean stopTimeshift()`	
Description	Stops rendering in timeshifted mode the broadcast channel in the video/broadcast object and, if applicable, plays the current broadcast from the live point and stops timeshifting the broadcast. The terminal may release all resources that were used to support timeshifted rendering of the broadcast. Returns true if the time-shifted broadcast was successfully stopped and false otherwise. If the video/broadcast object is currently not rendering a timeshifted channel, the terminal shall ignore the request to stop the timeshift and shall return the value false.

In addition to these methods, the terminal shall support an additional optional attribute "`offset`" on the `setChannel(Channel channel, Boolean trickplay, String contentAccessDescriptorURL)` method of the video/broadcast object as defined in OIPF DAE [1] section 7.13.1.3, if the terminal has indicated support for scheduled content over IP by defining one or more `ID_IPTV_*` values as part of the transport attribute of the `<video_broadcast>` element in the capability description.

```
void setChannel( Channel channel, Boolean trickplay,
                 String contentAccessDescriptorURL, Integer offset )
```

Description	Requests the terminal to switch a (logical or physical) tuner to the specified channel and render the received broadcast content in the area of the browser allocated for the video/broadcast object, as specified by the `setChannel(Channel channel, Boolean trickPlay, String contentAccessDescriptorURL)` method in OIPF DAE [1] section 7.13.1.3. The additional `offset` attribute optionally specifies the desired offset with respect to the live broadcast in number of seconds from which the terminal should start playback immediately after the channel switch (whereby `offset` is given as a negative value for seeking to a time in the past). If an terminal cannot start playback from the desired position, as indicated by the specified offset (e.g. because the terminal did not, or could not, record the specified channel prior to the call to `setChannel`), if the specified offset is '0', or if the offset is not specified, the terminal shall start playback from the live position after the specified channel switch.	
Arguments	*channel*	As defined for method `setChannel()` in OIPF DAE [1] section 7.13.1.3.
	trickplay	Optional flag as defined for method `setChannel()` in OIPF DAE [1] section 7.13.1.3.
	contentAccessDescriptorURL	Optional attribute as defined for method `setChannel()` in OIPF DAE [1] section 7.13.1.3.
	offset	The optional `offset` attribute may be used to specify the desired offset with respect to the live broadcast in number of seconds from which the terminal should start playback immediately after the channel switch (whereby `offset` is given as a negative value for seeking to a time in the past).

A.2.4.7.5 Events

For the intrinsic events "`onRecordingEvent`", "`onPlaySpeedChanged`" and "`onPlayPositionChanged`", corresponding DOM level 2 events shall be generated, in the following manner:

Intrinsic event	Corresponding DOM 2 event	DOM 2 Event properties
`onRecordingEvent`	`RecordingEvent`	Bubbles: No Cancelable: No Context Info: `state`, `error`, `recordingId`
`onPlaySpeedChanged`	`PlaySpeedChanged`	Bubbles: No Cancelable: No Context Info: `speed`
`onPlayPositionChanged`	`PlayPositionChanged`	Bubbles: No Cancelable: No Context Info: `position`
`onPlaySpeedsArrayChanged`	`PlaySpeedsArrayChanged`	Bubbles: No Cancelable: No Context Info: None

Note: the DOM 2 events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving these events during the bubbling or the capturing phase. Applications that use DOM 2 event handlers shall call the `addEventListener()` method on the video/broadcast object itself. The third parameter of `addEventListener`, i.e. "`useCapture`", will be ignored.

A.2.4.8 Extensions to video/broadcast for recording

A.2.4.8.1 General

If a terminal has indicated support in its capability description for recording functionality (i.e. by giving value `true` to the `<recording>` element as specified in OIPF DAE [1] section 9.3.3), the terminal shall support the following additional constants, properties and methods on the video/broadcast object, in order to start a recording of the current broadcast.

Note that this functionality is subject to the security model as specified in OIPF DAE [1] section 10.1.

The recording functionality is subject to the state transitions represented in the state diagram in **Error! Reference source not found.**.

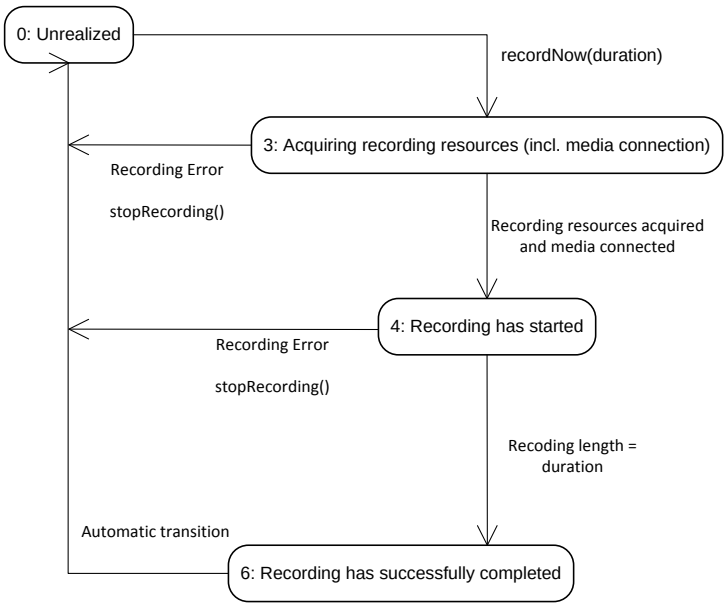


Figure 38: PVR States for recordNow using video/broadcast (normative)

Note that when the user switches to another channel whilst the current channel is being recorded using `recordNow()` or the video/broadcast object gets destroyed, the conflict resolution and the release of resources is implementation dependent. The terminal may report a recording error using a `RecordingEvent` with value 0 ("Unrealized") for argument `state` and with value 2 ("Tuner conflict") for argument `error` in that case.

A.2.4.8.2 Properties

`readonly Integer recordingState`

Returns the state of the terminal's timeshift and recordNow functionality for the channel shown in the video/broadcast object. One of:

Value	Description
0	Unrealized: user/application has not requested timeshift or recordNow functionality for the channel shown. No timeshift or recording resources are claimed in this state.
1	Value not used
2	Value not used
3	Acquiring recording resources (for example, space on the media storage device).
4	Recording has started.
5	Value not used
6	Recording has successfully completed.

```
function onRecordingEvent( Integer state, Integer error, String recordingId )
```

This function is the DOM 0 event handler for notification of state changes of the recording functionality. The specified function is called with the following arguments:

- Integer state - The current state of the recording. One of:

Value	Description
0	Unrealized: user/application has not requested timeshift or recordNow functionality for the channel shown. No timeshift or recording resources are claimed in this state.
1	Value not used
2	Value not used
3	Acquiring recording resources (for example, space on the media storage device).
4	Recording has started.
5	Value not used
6	Recording has successfully completed.

- Integer error - The current state of the recording. One of:

Value	Description
0	The recording sub-system is unable to record due to resource limitations.
1	There is insufficient storage space available. (Some of the recording may be available).
2	Value not used
3	Recording not allowed due to DRM restrictions.
4	Recording has stopped before completion due to unknown (probably hardware) failure.
5	Value not used
6	Recording has successfully completed.

  If no error has occurred, this argument shall take the value undefined.

- String recordingId - The identifier of the recording to which this event refers, This shall be equal to the value of the id property for the affected recording, if the event is associated with a specific recording. This shall be undefined when the value of state is 0.

A.2.4.8.3 Methods

```
String recordNow( Integer duration )
```

Description	Starts recording the broadcast currently rendered in the video/broadcast object. If the terminal has buffered the broadcasted content, the recording starts from the current playback position in the buffer, otherwise start recording the broadcast stream as soon as possible after the recording resources have been acquired. The specified duration is used by the terminal to determine the minimum duration of the recording in seconds from the current starting point. Calling `recordNow()` while the broadcast that is currently rendered in the video/broadcast object is already being recorded, shall have no effect on the recording and shall return the value `null`. In other cases, this method returns a `String` value representing a unique identifier to identify the recording. If the terminal provides recording management functionality through the APIs defined in OIPF DAE [1] section 7.10.4, this shall be the value of the `id` property of the associated Recording object defined in OIPF DAE [1] section 7.10.5. The terminal shall guarantee that recording identifiers are unique in relation to download identifiers and `CODAsset` identifiers. The method returns `undefined` if the given argument is not accepted to trigger a recording. If the terminal supports metadata processing in the terminal, the fields of the resulting `Recording` object may be populated using metadata retrieved by the terminal. Otherwise, the values of these fields shall be implementation-dependent.	
Arguments	*duration*	The minimum duration of the recording in seconds. A value of -1 indicates that the recording should continue until `stopRecording()` is called, storage space is exhausted, or an error occurs. In this case it is essential that `stopRecording()` is called later.

`void stopRecording()`	
Description	Stops the current recording started by recordNow().

A.2.5 Extensions to the A/V Control object

A.2.5.1 New queue method

The following method shall be added to the A/V Control embedded object.

```
Boolean queue(String url)
```

Description	Queue the media referred to by `url` for playback after the current media item has finished playing. If a media item is already queued, `url` will not be queued for playback and this method will return false. If the item is queued successfully, this method returns true. If no media is currently playing, the queued item will be played immediately. If `url` is null, any currently queued item will be removed from the queue and this method will return true. If an A/V Control object is an audio object (as defined by clause 7.14 of the OIPF DAE specification [1]) then queued media items shall only contain audio. Otherwise, if an A/V Control object is a video object | then queued media items shall always contain video and may also contain audio and other media components. When the current media item has finished playing, the A/V Control object shall transition to the finished state, update the value of the `data` property with the URL of the queued media item and automatically start playback of the queued media item. The A/V Control object may transition to the connecting or buffering states before entering the playing state when the queued media item is being presented. Implementations may pre-buffer data from the queued URL before the current media item has finished playing in order to reduce the delay between items. Play speed is not affected by transitioning between the current and queued media item. To avoid race conditions when queueing multiple items for playback, applications should wait for the currently queued item to begin playback before queuing subsequent items, e.g. by queueing the subsequent item when the A/V Control object transitions to the connecting, buffering or playing state for the currently queued item.	
Arguments	`url`	The media item to be queued, or null to remove the currently-queued item.

Calling `stop()`, modifying the `data` and/or `type` property or entering the error state shall cause any queued media item to be discarded.

Play control keys (OK, play, stop, pause, fast forward, fast rewind and other trick play keys) shall not be handled by the A/V Control object and no action shall be taken by the terminal for these keys when they have been requested by an application. DOM 2 events shall be generated for these keys whether the A/V Control object is focused or not.

The timing of automatic transitions from the error state to the stopped state is implementation dependent; applications should not rely on the A/V Control object remaining in the error state after an error has occurred and should listen for play state change events in order to detect errors.

If the AVControl object's `play()` method returns true then at least one play state change event shall be generated.

The `error` property shall be available in the stopped state. After an automatic transition from the error state to the stopped state, the value of the `error` property shall be preserved.

The following value shall be added to the list of valid values for the error property:

- undefined - no error has occurred;
- 7 - content blocked due to parental control.

A.2.5.2 State machine and related changes

This clause describes a set of changes to the state machine for the A/V Control object defined in clause 7.14.1.1 of the OIPF DAE specification [1]:

- An A/V Control object with a CSS rule of `display:none` shall not be loaded and hence shall not be decoding audio or video.

A.2.5.3 Support for TTML subtitles

The following extensions shall apply to support TTML subtitles as defined by clause 7.3.1.5. The `<object>` element of an A/V Control object shall contain a `<param>` element for each subtitle component that is carried out-of-band.

The attributes of the <param> element shall have the following values.

<table>
<tr><th>Attribute name</th><th>Attribute value</th></tr>
<tr><td>name</td><td>"subtitles" or "captions"</td></tr>
<tr><td>value</td><td>Shall contain a list of key:value tuples separated by space.
The following keys shall be supported:
<table>
<tr><th>Key</th><th>Value</th></tr>
<tr><td>srclang</td><td>the language of the subtitles. This value should be the same as defined by the xml:lang attribute in clause 3 of [43].</td></tr>
<tr><td>src</td><td>The HTTP URL to the TTML document. The URL shall use percent encoding as defined in IETF RFC 3986 [27].</td></tr>
<tr><td>Label</td><td>a textual representation for the subtitle track.</td></tr>
</table>
</td></tr>
</table>

The terminal shall support the component selection API defined in clause 7.14.4.1 of OIPF DAE [1] for in-band and out-of-band delivered TTML subtitles as defined in clause 7.3.1.5. The following mappings for the properties of the AVSubtitleComponent class as defined in clause A.2.13 shall apply.

property	Value
encoding	Shall have the value "application/ttml+xml"
language	The value defined by the 'srclang' key used in the <param> element.
hearingImpaired	Shall be true if the value of the 'name' attribute is 'captions', false otherwise.
label	The value defined by the 'label' key used in the <param> element.

EXAMPLE:

```
<object type='video/mp4' data='http://mycdn.de/video.mp4'>
<param name='subtitles ' value='srclang:de src: http%3A%2F%2Fmycdn.de%2Fsubtitles_de.ttml' />
<param name='captions ' value='srclang:en src:
    http%3A%2F%2Fmycdn.de%2Fsubtitles_hearing_impaired.ttml' />
```

A.2.5.4 Support for media synchronization with subtitle-only streams

The media synchronization capabilities of the terminal as defined in clause 10.2.8 allow for delivery of subtitles as broadband streams that are re-synchronized in the terminal with another stream, either broadcast or broadband.

The terminal shall support the rendering of subtitles when carried in a separate subtitle-only stream using the A/V Control object if the type of the subtitle-only stream is supported by the terminal for media synchronization. The requirements on terminals for supporting subtitle-only streams are defined in clause 10.2.8.4. In any case, the following restrictions shall apply:

- For DVB Subtitles:
  - the video component is provided by the broadcast service;
  - the type attribute of the A/V Control object is "video/mpeg";
  - the subtitle stream is delivered with a Content Type of "image/vnd.dvb.subtitle"; and
  - the subtitle stream is a single programme transport stream.
- For TTML subtitles:
  - the video component is provided by the broadcast service;
  - EITHER:
    - the type attribute of the A/V Control object is "application/ttml+xml";
    - the subtitle stream is delivered with a Content Type of "application/ttml+xml"; and

        - the subtitle stream is a single XML document.
    - OR:
        - the subtitle stream is delivered using MPEG-DASH; and
        - multi-stream media synchronization is used, as specified in clause 10.2.8.
- For Teletext subtitles (if supported):
    - the video component is provided by the broadcast service;
    - the type attribute of the A/V Control object is "`video/mpeg`";
    - the subtitle stream is delivered with a Content Type "`text/vnd.dvb.teletext`"; and
    - the subtitle stream is a single programme transport stream.

The terminal shall render the subtitle stream on the subtitles plane with scaling and positioning defined by the subtitle object element and not by any video element that it is synchronized with.

A.2.5.5 Using an A/V Control object to play downloaded content

Clause 7.14.1.3 "Using an A/V Control object to play downloaded content" shall be modified by the addition of the text shown underlined.

If an A/V Control object is used to play content that has been downloaded and stored on the OITF (by using method `setSource()` as defined in clause 7.14.7) then the following holds:

- if the download was triggered using `registerDownloadURL()` with a `contentType` other than "`application/vnd.oipf.ContentAccessDownload+xml`" or the download was triggered using a Content Access Download Descriptor with `<TransferType>` value "`playable_download`" as defined in clause E.1, then:
    - if the `play()` method is called before sufficient data has been download to initiate playback, then the play state of the A/V Control object SHALL be set to 6 ('error') with a detailed error code of 5 ("content not available").

A.2.5.6 Extension to PlayStateChange event

In clause 7.14.1, the description of the 'onPlayStateChange' property is replaced with the following;

> The function that is called when the play state of the A/V control object object changes for any reason.
>
> The specified function shall be called with the argument state. This argument is defined as follows:
>
> - Number state – the new state of the A/V control object. Valid values are given in the definition of the playState property [Req. 5.7.1.f].

A.2.5.7 Other modifications to the A/V Control object

In clause 7.14.3.1, the definition of the property `onPlayPositionChanged( Integer position )` is changed as shown;

The function that is called when change occurs in the play position of a channel due to the use of ~~trick play functions~~ random access.

A.2.6 HTML Profile

A.2.6.1 Void

A.2.6.2 MIME type and DOCTYPE

All HTML and XHTML documents of an HbbTV® application should use the DOCTYPE defined for HTML5 in the HTML5 specification as profiled by the OIPF Web Standards TV Profile [i.6].

Terminals shall support the DOCTYPE defined for HTML5 in the HTML5 specification as profiled by the OIPF Web Standards TV Profile [i.6] and shall also support the following DOCTYPEs in order to run applications authored for previous versions of the present document.

- The Strict XHTML doctype (for documents that are conformant with the subset of the XHTML 1.0 Strict DTD defined in the present document).
- The Transitional XHTML doctype (for documents that are conformant with the subset of the XHTML 1.0 Transitional DTD defined in the present document).
- The following "doctype" declaration:

```
<!DOCTYPE html PUBLIC "-//HbbTV//1.1.1//EN" "http://www.hbbtv.org/dtd/HbbTV-1.1.1.dtd">
```

- The following "doctype" declaration:

```
<!DOCTYPE html PUBLIC "-//HbbTV//1.2.1//EN" "http://www.hbbtv.org/dtd/HbbTV-1.2.1.dtd">
```

Terminals are not required to load or run documents which do not include one of the doctype declarations defined or referenced above.

When loading an HbbTV® document, a terminal shall not use the suffix from the filename to determine the document type.

All HTML documents of an HbbTV® application should be served with one of the MIME types defined for HTML5.

Terminals shall support the MIME types defined for HTML5 and shall also support the following MIME type in order to run applications authored for previous versions of the present document:

`application/vnd.hbbtv.xhtml+xml`

Content served with the `application/vnd.hbbtv.xhtml+xml` Content-Type shall be parsed using the rules in the HTML5 specification as profiled by the OIPF Web Standards TV Profile [i.6] for the content type:

`application/xhtml+xml`

Terminals are not required to load or run documents which are served using HTTP with a MIME type other than those defined or referenced above. Terminals shall use the DOCTYPE to determine the type of documents loaded from a carousel or CICAM.

NOTE: Different requirements apply for the MIME type that serves as an application type identifier in the XML AIT. See clause 7.2.3.2.

A.2.6.3 Void

A.2.6.4 Browser History

The terminal should not offer a history UI for HbbTV® applications.

The behaviour of the history mechanism when an HbbTV® application transitions between broadcast-independent and broadcast-related (or vice-versa) is outside the scope of the present document. Implementations may record and reproduce these transitions when the history mechanism is used but are not required to do so.

A.2.6.5 Attribute reflection for visual embedded objects

The IDL attributes of an `<object>` element representing an A/V Control or video/broadcast object shall reflect the element's content attributes of the same names respectively, as defined in clauses 2.7.1, 4.7.4 and 4.7.16 of the HTML5 Recommendation [54].

NOTE: This means that the attributes '`data`', '`type`', '`name`', '`width`', and '`height`' can be set and read either by accessing the object element's JavaScript properties of the same names, or by invoking the `<object>` element's `setAttribute()`/`getAttribute()` methods.

A.2.7 Extensions to the oipfObjectFactory object

The `oipfObjectFactory` object as defined in clause 7.1 of the OIPF DAE specification [1] shall be extended by the methods defined in this clause.

`MediaSynchroniser createMediaSynchroniser ()`	
Description	Creates a new `MediaSynchroniser` embedded object.

`Object createCSManager ()`	
Description	Creates a new `HbbTVCSManager` embedded object.

The `isObjectSupported()` method shall be extended to support the following MIME types for querying support of new functionality defined in the present document:

- `application/hbbtvMediaSynchroniser`
- `application/hbbtvCSManager`

A.2.8 Void

A.2.9 Access to EIT Schedule Information

The Metadata APIs listed in Table A.1 of the present document shall allow access to DVB-SI EIT event schedule information for the actual transport stream and for the other transport streams (as defined in EN 300 468 [16]) that are carried on the transport stream of the currently selected broadcast service.

The terminal shall use EIT-present/following information and, if present, EIT-schedule information. If both EIT-schedule and EIT-present/following information are present, it is implementation dependent which shall be used in cases where there are conflicts.

A.2.10 Correction to the application/oipfDownloadManager object

In clause 7.4.3.2 of the OIPF DAE specification [1], the definition of the `allocated` property shall be superseded by the following definition.

`readonly Integer allocated`
Returns the space (in megabytes) allocated for all downloads registered by this application and by applications from the same `organisation_id` as this application. It shall be calculated as the sum of the `totalSize` properties of all the download objects registered by any application from the same `organisation_id` as the calling application.

A.2.11 Extensions to the Download class

This class shall be extended with the following additional property.

`readonly Number errorLevel`
A representation of the quantity of detected but uncorrected errors in a file downloaded using FDP (as defined in annex H). The value of this property shall be calculated as the number of erroneous or missing File Segments divided by the total number of File Segments. A value of zero indicates that the downloaded file has no errors. If `state` does not equal 1, or if the file is not downloaded using FDP, then the value of this property is not defined.

A.2.12 HTML5 media element mapping

A.2.12.1 Inband VideoTracks, AudioTracks and TextTracks

The following shall apply when an HTML5 media element is presenting content whose system format is the MPEG-2 transport stream format:

- A `VideoTrack` object shall be created for each elementary stream in the transport stream where the '`stream_type`' is "0x01", "0x02", "0x1B", or "0x24". The order of `VideoTrack` objects in the `VideoTrackList` shall be the same as the order of the corresponding elementary stream in the PMT.
- Audio elementary streams in the transport stream shall be recognised based on meeting one of the following criteria:
  - the '`stream_type`' is "0x03", "0x04", or "0x11";
  - it is an AC-3 audio component as identified by an `AC-3_descriptor` (as defined in EN 300 468 [16]) in the 'Elementary Stream Descriptors' in the PMT entry for a stream with a '`stream_type`' of "0x06";
  - it is an Enhanced AC-3 audio component as identified by an `enhanced_ac-3_descriptor` (as defined in EN 300 468 [16]) in the 'Elementary Stream Descriptors' in the PMT entry for a stream with a '`stream_type`' of "0x06";
  - it is a DTS® audio component as identified by a `DTS_audio_stream_descriptor` (as defined in EN 300 468 [16]) in the 'Elementary Stream Descriptors' in the PMT entry for a stream with a '`stream_type`' of "0x06";
  - it is a DTS-HD® audio component as identified by a `DTS-HD_audio_stream_descriptor` (as defined in EN 300 468 [16]) in the 'Elementary Stream Descriptors' in the PMT entry for a stream with a 'stream_type' of "0x06".

  An `AudioTrack` object shall be created for each audio elementary stream that does not have a `supplementary_audio_descriptor` (as defined in EN 300 468 [16]) with an audio purpose of "Audio description (receiver-mix)".

The following shall apply for each audio elementary stream that has a `supplementary_audio_descriptor` (as defined in EN 300 468 [16]) with an audio purpose of "Audio description (receiver-mix)":

- An `AudioTrack` object shall be created for each permitted combination of this audio elementary stream with another audio elementary stream as defined in clause J.2 of EN 300 468 [16]. Enabling such an AudioTrack object shall result in the combination being presented.

- If the HbbTV® terminal can present the stream in isolation, it shall also create an `AudioTrack` object for this audio component outside of a combination.

The order of `AudioTrack` objects in the `AudioTrackList` shall be the same as the order of the corresponding elementary stream in the PMT.

- A `TextTrack` object shall be created for each elementary stream in the transport stream that meets one of the following criteria:

  - it is a DVB subtitle component as identified by a `subtitling_descriptor` (as defined in EN 300 468 [16]) in the 'Elementary Stream Descriptors' in the PMT entry for a stream with a '`stream_type`' of "0x06"

  - it is an ITU-R System B Teletext component as identified by a `teletext_descriptor` (as defined in EN 300 468 [16]) in the 'Elementary Stream Descriptors' in the PMT entry for a stream with a '`stream_type`' of "0x06"

  The order of `TextTrack` objects in the `TextTrackList` shall be the same as the order of the corresponding elementary stream in the PMT.

The following shall apply when an HTML5 media element is presenting content whose system format is the ISO BMFF:

- A `VideoTrack` object shall be created for each track in the ISOBMFF file whose 'handler_type' is 'vide'. The order of `VideoTrack` objects in the `VideoTrackList` shall be the same as the order of the corresponding 'trak' boxes in the 'moov' box.

- A `AudioTrack` object shall be created for each track in the ISOBMFF file whose 'handler_type' is 'soun'. The order of `AudioTrack` objects in the `AudioTrackList` shall be the same as the order of the corresponding 'trak' boxes in the 'moov' box.

- A `TextTrack` object shall be created for each track in the ISOBMFF file whose 'handler_type' is either 'subt' or 'text' and whose `SampleEntryFormat` is `XMLSubtitleSampleEntry` as defined in ISO/IEC 14496-30 [52]. The order of `TextTrack` objects in the `TextTrackList` shall be the same as the order of the corresponding 'trak' boxes in the 'moov' box.

The following shall apply when an HTML5 media element is presenting content whose system format is MPEG DASH:

- A `VideoTrack` object shall be created for each video Adaptation Set in the MPD. The order of `VideoTrack` objects in the `VideoTrackList` shall be the same as the order that the corresponding Adaptation Sets are in the MPD.

- A `AudioTrack` object shall be created for each audio Adaptation Set in the MPD. The order of `AudioTrack` objects in the `AudioTrackList` shall be the same as the order that the corresponding Adaptation Sets are in the MPD.

- A `TextTrack` object shall be created for each Adaptation Set in the MPD carrying EBU-TT-D TTML data as defined in the DVB DASH profile TS 103 285 [45]. The order of `TextTrack` objects in the `TextTrackList` shall be the same as the order that the corresponding Adaptation Sets are in the MPD.

- A `TextTrack` object shall be created for each event stream as defined in clause 9.3.2.2 of the present document.

NOTE 1: The HTML5 specification requires that the `VideoTrack`/`AudioTrack`/`TextTrack` objects will have been created by the time the `readyState` attribute of the media element enters the `HAVE_METADATA` state.

NOTE 2: It is intentional that creation of `VideoTrack`, `AudioTrack` and `TextTrack` objects is required for tracks that the terminal cannot decode.

A.2.12.2 Out-of-band text tracks

Independent of the system format, terminals shall create `TextTrack` objects for HTML `<track>` elements representing TTML subtitle components that are carried out-of-band (see clause 7.3.1.5) when represented by a `<track>` element in an HTML document. E.g.:

```
<video>
    <source src='http://mycdn.de/video.mp4' type='video/mp4'>
    <track kind='subtitles' srclang='de' label='German for the English'
           src='http://mycdn.de/subtitles_de.ttml' />
    <track kind='subtitles' srclang='de' label='German for the hard of hearing'
           src='http://mycdn.de/subtitles_de2.ttml' />
    <track kind='captions' srclang='en' src='http://mycdn.de/subtitles_hearing_impaired.ttml' />
</video>
```

A.2.12.3 Modifications to clause 8.4.6

The definition of the value of the `kind` property of a `TextTrack` in the MPEG DASH system layer shall be replaced with the following:

- "captions": if (role is "main" AND the MPD contains an audio Adaptation Set with role "main" and the same language as the subtitle track AND an accessibility descriptor is present with the schemeIdUri set to "`urn:tva:metadata:cs:AudioPurposeCS:2007`" and a value 2 [for the hard of hearing]) OR (role is "commentary");
- "subtitles": if (role is "alternate") OR (role is "main" AND no accessibility scheme is specified);
- "metadata": otherwise.

The definition of the `kind` property of a `VideoTrack` and `AudioTrack` in the MPEG DASH system layer shall be replaced with the following:

Given a role scheme of "`urn:mpeg:dash:role:2011`", determine the `kind` attribute from the value of the role descriptors in the `<AdaptationSet>` element.

- "alternative": if the role is "alternate" but not also "main" or "commentary", or "dub";
- "captions": if the role is "caption" and also "main";
- "descriptions": if the role is "description" and also "supplementary";
- "main": if the role is "main" but not also "caption", "subtitle", or "dub";
- "main-desc": if the role is "main" and also "description";
- "sign": not used;
- "subtitles": if the role is "subtitle" and also "main";
- "translation": if the role is "dub" and also "main";
- "commentary": if the role is "commentary" but not also "main";
- "": otherwise.

In the table defining "the mapping that SHALL be used between the HTML5 AudioTrack and the MPEG-2 transport stream and ISO BMFF system layers" contained in the OIPF DAE specification [1], the requirement for the language attribute does not apply for MPEG DASH. Where an `AudioTrack` corresponds to an MPEG DASH adaptation set, the value of the `AudioTrack.language` attribute shall be the value of the `lang` attribute in the MPD.

A.2.13 Extensions to the AVSubtitleComponent class

The following property shall be added to the `AVSubtitleComponent` class.

`readonly String label`
The label identifies a component by a human readable short description.

A.2.14 Modifications to clause H.2 "Interaction with the video/broadcast and A/V Control objects"

Clause H.2 of the OIPF DAE specification [1] defines the scaling and clipping of video when not in full screen mode as follows.

When the video/broadcast object or A/V Control object is not in "full-screen mode", any video being presented shall be scaled and positioned in the following way:

- if the video/broadcast object has the same aspect ratio as the video the four corners of the video shall match exactly the corners of the video/broadcast object;
- otherwise the video shall be scaled such that one side of the video fills the video/broadcast object fully without cropping the picture. The aspect ratio shall be preserved. Along the side where the video is shorter than the video/broadcast object, the video shall be centered. The area of the video plane not containing video shall be opaque black.

The above text shall not be interpreted as preventing video that is being presented by an object not in full screen mode from being scaled and/or cropped where this is indicated in the video stream. Specifically:

- If the video indicates that only part of the frame is of interest and the remainder can be cropped (e.g. AFD or Bar Data as defined in TS 101 154 [14] or the 'default display window' from HEVC), processing of that indication is permitted for objects not in full screen mode.

NOTE: The present document is intentionally silent on whether a terminal is required to act on AFD and Bar Data. In practice terminals decoding video using typical digital TV silicon will likely act on it but terminals decoding video using other silicon or software may not act on it. Applications wishing to ensure that video is scaled and/or cropped on all terminals should handle this in the application and not rely on the video decoder to do it.

- If the video is encoded at one resolution but is indicated as being required to be displayed at a different resolution then that processing (i.e. scaling the video to the display resolution) is permitted for objects not in full screen mode. For example, TS 101 154 [14] defines how broadcast video can be transmitted with reduced horizontal luminance resolution but to be up-sampled to full-screen size in the terminal. For example, with MPEG DASH ISO/IEC 23009-1 [29], the nominal display size in square pixels after decoding, AVC cropping, and rescaling is indicated by the width and height values in track header box. The actual encoded resolution will differ between Representations.

If the video is cropped or scaled as indicated by either of the indications referred to above then the original requirements defined in clause H.2 of the OIPF DAE specification [1] (quoted above) shall apply to the video after that processing has been performed.

A.2.15 Extensions to the OIPF-defined capability negotiation mechanism

The following schema is an extension of the schema defined by annex F of the OIPF DAE specification [1]:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:hbbtv="urn:hbbtv:config:oitf:oitfCapabilities:2014-1"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="urn:hbbtv:config:oitf:oitfCapabilities:2014-1"
           xmlns:oipf="urn:oipf:config:oitf:oitfCapabilities:2011-1"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
    <xs:import namespace="urn:oipf:config:oitf:oitfCapabilities:2011-1"
               schemaLocation="oipf\config-oitf-oitfCapabilities.xsd"/>
    <xs:import schemaLocation="oipf\imports/ce-html-profiles-1-0.xsd"/>
    <xs:element name="profilelist" type="hbbtv:profileListType"/>
    <xs:complexType name="profileListType">
        <xs:sequence>
            <xs:element name="ui_profile" type="hbbtv:uiProfileType" maxOccurs="unbounded"/>
            <xs:element name="audio_profile" type="hbbtv:audioProfileType"
                        minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="video_profile" type="hbbtv:videoProfileType"
                        minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
```

```
    </xs:complexType>
    <xs:complexType name="videoProfileType">
        <xs:complexContent>
            <xs:extension base="oipf:videoProfileType">
                <xs:attribute name="sync_tl" type="hbbtv:sync_tl_type" use="optional"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="audioProfileType">
        <xs:complexContent>
            <xs:extension base="oipf:audioProfileType">
                <xs:attribute name="sync_tl" type="hbbtv:sync_tl_type" use="optional"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="uiProfileType">
        <xs:sequence>
            <xs:element name="ext" type="hbbtv:uiExtensionType" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>
    <xs:complexType name="uiExtensionType">
        <xs:complexContent>
            <xs:extension base="uiExtensionType">
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                    <xs:element name="video_broadcast" type="oipf:videoBroadcastType"
                                minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="overlaylocaltuner" type="oipf:overlayType"/>
                    <xs:element name="overlayIPbroadcast" type="oipf:overlayType"/>
                    <xs:element name="recording" type="oipf:pvrType"/>
                    <xs:element name="parentalcontrol" type="oipf:parentalControlType"/>
                    <xs:element name="extendedAVControl" type="xs:boolean"/>
                    <xs:element name="clientMetadata" type="oipf:metadataType"/>
                    <xs:element name="configurationChanges" type="xs:boolean"/>
                    <xs:element name="communicationServices" type="xs:boolean"/>
                    <xs:element name="presenceMessaging" type="xs:boolean"/>
                    <xs:element name="drm" type="oipf:drmType" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="remote_diagnostics" type="xs:boolean"/>
                    <xs:element name="pollingNotifications" type="xs:boolean"/>
                    <xs:element name="mdtf" type="xs:boolean"/>
                    <xs:element name="widgets" type="xs:boolean"/>
                    <xs:element name="html5_media" type="xs:boolean"/>
                    <xs:element name="remoteControlFunction" type="xs:boolean"/>
                    <xs:element name="wakeupApplication" type="xs:boolean"/>
                    <xs:element name="wakeupOITF" type="xs:boolean"/>
                    <xs:element name="hibernateMode" type="xs:boolean"/>
                    <xs:element name="telephony_services" type="oipf:telephonyServicesType"/>
                    <xs:element name="playbackControl" type="oipf:playbackType"/>
                    <xs:element name="temporalClipping" type="oipf:hasCapability"/>
                    <xs:element name="graphicsPerformance" type="hbbtv:graphicsPerformanceType"/>
                    <xs:any namespace="##other"/>
                </xs:choice>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:simpleType name="sync_tl_type">
        <xs:list itemType="hbbtv:sync_tl_values_type"/>
    </xs:simpleType>
    <xs:simpleType name="sync_tl_values_type">
        <xs:restriction base="xs:string">
            <xs:enumeration value="pts"/>
            <xs:enumeration value="ct"/>
            <xs:enumeration value="temi"/>
            <xs:enumeration value="dash_pr"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="graphicsPerformanceType">
        <xs:attribute name="level" type="xs:string"/>
    </xs:complexType>
</xs:schema>
```

This schema extends the `<video_profile>` and `<audio_profile>` elements with an optional attribute "`sync_tl`". When present, the value of this attribute shall be a space separated concatenation of one or more option strings denoting types of timelines. The values of the option strings and the types of timelines that they correspond to are defined in Table A.4. The types of timeline listed in Table A.4 are defined in clause 13.4.

Table A.4: Values of sync_tl attribute option strings

option string in sync_tl attribute	Type of timeline denoted by the option string
pts	MPEG-TS Presentation Timestamps
ct	ISOBMFF Composition Time
temi	MPEG-TS Timed External Media Information
dash_pr	MPEG DASH Period-Relative Timeline

The value of the `sync_tl` attribute (when present in an element describing an audio or video profile) indicates the types of timeline that the terminal supports for use with the `MediaSynchroniser` API for a media object that represents:

- any broadband stream matching the profile; or
- any broadband stream matching the profile but where some, but not all, of the audio, video and subtitle components defined in the profile are not present in the stream.

NOTE 1: If no types of timeline are indicated as supported for a given profile of broadband stream, then by implication it is not possible to use that stream with the `MediaSynchroniser` API and therefore by implication not possible to use that stream for multi-stream or inter-device synchronization.

This schema extends the set of capabilities that can be indicated by a terminal to include graphics performance by adding a new `<graphicsPerformance>` element that can appear within an `<ext>` element.

NOTE 2: Due to limitations of XML schema syntax and semantics, the schema above repeats the definitions of elements permitted within an `<ext>` element that are defined in the OIPF DAE specification [1]. The meaning and definition of these elements are unchanged.

The following semantics shall apply for the `<graphicsPerformance>` element.

The `<graphicsPerformance>` element indicates that the terminal declares its graphics performance. This element has the following attribute:

- attribute "`level`": if the `<graphicsPerformance>` element is present, this attribute SHALL include a non-empty space separated list of the graphics performance levels with which the terminal complies, encoded as one or more URNs. For terminals conforming to a performance level as defined in clause 12.1 of the OIPF DAE specification [1] and modified by annex A of the present document, the value of the `level` attribute shall be the corresponding URN specified in Table A.5 below.

Table A.5: HbbTV® Graphics Performance Levels

Performance Level	URN
Level 1	`urn:hbbtv:graphics:performance:level1`
Level 2	`urn:hbbtv:graphics:performance:level2`

Terminals that do not comply with any of the graphics performance levels referred to above shall not include the `<graphicsPerformance>` element.

A.2.16 Graphics performance

The following modifications to clause 12.1 of the OIPF DAE specification [1] shall apply:

- Clause 12.1.7 is not included in the present document.
- The second paragraph of clause 12.1.3 is modified by the addition of the underlined text:

  "Values in this table indicate the number of elements of the specified target being animated simultaneously and updated at 25 Hz . The number is expressed as a power of 2, i.e. a value of 3 SHALL mean 4 simultaneous animations, a value of 5 SHALL mean 16 simultaneous animations."

- Table 17 "Minimum 2D graphics performance" of [1] is modified as follows:

Target for the CSS Property	CSS Property being animated	~~Test~~	Level 1	Level 2
Frame	background-color	~~2d/frame-color~~	3	5
	background-color, opacity	~~2d/frame-color-alpha~~	3	5
	left, top	~~2d/frame-left-top~~	3	5
	opacity	~~2d/frame-opacity~~	3	5
	transform: rotate	~~2d/frame-rotate~~	No requirement	5
	transform: scale	~~2d/frame-scale~~	3	5
	transform: skew	~~2d/frame-skew~~	No requirement	5
	transform: matrix	~~2d/frame-matrix~~	No requirement	5
	border-radius	~~2d/frame-border-radius~~	3	5
	width, height	~~2d/frame-width-height~~	3	5
	linear-gradient	~~2d/frame-linear-gradient~~	3	5
Image	left, top	~~2d/image-top-left~~	3	5
	opacity	~~2d/image-opacity~~	3	5
	transform: rotate	~~2d/image-rotate~~	No requirement	5
	transform: scale	~~2d/image-scale~~	3	5
	transform: skew	~~2d/image-skew~~	No requirement	5
	transform: matrix	~~2d/image-matrix~~	No requirement	5
Text	left, top	~~2d/text-left-top~~	3	5
	opacity	~~2d/text-opacity~~	3	5
	transform: rotate	~~2d/text-rotate~~	No requirement	5
	transform: scale	~~2d/text-scale~~	3	5
	transform: skew	~~2d/text-skew~~	No requirement	5
	text-shadow	~~2d/text-emboss~~	3	5

A.2.17 Notification of change of components

The video/broadcast and A/V Control object shall be extended with the following property.

`function onComponentChanged`
This function is called when there is a change in the set of components in the current stream, i.e. the set of all components that would be returned by the `getComponents()` method. The specified function is called with one argument: • `Integer componentType` - The type of component for which there has been a change in the current stream, as represented by one of the constant values listed in clause 7.16.5.1.1 of the OIPF DAE specification [1]. If there has been a change for more than one type of component, this argument will take the value undefined.

The video/broadcast and A/V Control object shall be extended with the following event.

Intrinsic event	Corresponding DOM 2 event	DOM 2 Event properties
onComponentChanged	ComponentChanged	Bubbles: No Cancelable: No Context Info: componentType

A.2.18 Clarification regarding the reserve() method of the application/oipfDownloadManager object

The following rules and clarifications shall apply to the `reserve()` method of the `application/oipfDownloadManager` embedded object.

In all cases, if a call to the `reserve()` method does not succeed (i.e. if the return status is other than `RESERVE_OK`), this call shall have no effect. Any prior reservation is kept unchanged.

A.2.19 Correction to the registerDownloadURL() method

In clause 7.4.1.1 of the OIPF DAE specification [1], the definition of the `registerDownloadURL()` method shall be superseded by the following definition.

<table>
<tr><td colspan="3"><code>String registerDownloadURL(String URL, String contentType,Date downloadStart,
Integer priority)</code></td></tr>
<tr><td>Description</td><td colspan="2">This method triggers the terminal to initiate a download of the content pointed to by the URL and the given content type.
The <code>contentType</code> argument SHALL reflect the expected type of content returned by the content server when connecting to the URL. The contentType can be used to evaluate if the content type is part of the list of accepted content types of the terminal. For example, if the terminal does not support content type "<code>video/MP2T</code>", then the <code>registerDownloadURL()</code> method could return undefined to indicate this to the application in advance of the download.

If contentType has value "<code>application/vnd.oipf.ContentAccessDownload+xml</code>", the method SHALL return a download identifier, after which the terminal SHALL immediately fetch the Content Access Download Descriptor. In cases where the Content Access Download Descriptor is not accepted by the terminal, the <code>state</code> property of the download shall be immediately changed to the value 8 ('Failed') and the <code>reason</code> property shall be set to 4 ('Other reason'). Otherwise, the same SHALL happen as if <code>registerDownload()</code> as defined in clause 4.6.3.1 with the given Content Access Download Descriptor as argument was called. The <code>downloadStart</code> argument only applies to the individual <code>Download</code> objects described by the Content Access Download Descriptor and SHALL NOT apply to the retrieval of the Content Access Download Descriptor itself.

Note that if the Content Access Download Descriptor contains multiple content items to be downloaded, the associated <code>Download</code> objects for each of these content items SHALL have the same value for the <code>id</code> property. The associated <code>Download</code> objects can be retrieved through the <code>createFilteredList()</code> method as defined in clause 7.4.3.3.
Returns a <code>String</code> value representing a unique identifier to identify the download, if the given arguments are acceptable by the terminal to trigger a download. If the terminal supports the <code>application/oipfDownloadManager</code> as specified in clause 7.4.3, this SHALL be the value of the <code>id</code> attribute of the associated <code>Download</code> object(s).

The terminal SHALL guarantee that download identifiers are unique in relation to recording identifiers and <code>CODAsset</code> identifiers.

The method returns <code>undefined</code> if the given arguments are not acceptable by the terminal to trigger a download.</td></tr>
<tr><td rowspan="4">Arguments</td><td><code>URL</code></td><td>The URL from which the content can be fetched.</td></tr>
<tr><td><code>contentType</code></td><td>The type of content referred to by the URL attribute. The <code>contentType</code> can be used to evaluate if the content type is part of the list of supported content types of the terminal.</td></tr>
<tr><td><code>downloadStart</code></td><td>Optional argument indicating the time at which the download should be started. If the argument is not included, or takes a value of null then the download should start as soon as possible.</td></tr>
<tr><td><code>priority</code></td><td>Optional argument indicating the relative priority of the download with respect to other downloads registered by the same organisation as the calling application. Higher values indicate a higher priority. If the argument is not included then a priority of 0 shall be assigned.</td></tr>
</table>

A.2.20 Extensions to the Configuration class

A.2.20.1 Extensions to Represent Subtitle Presentation

This class shall be extended with the following additional property.

`readonly Boolean subtitlesEnabled`
Shall be set to false if subtitles are disabled by the terminal and applications cannot enable subtitles using the component selection API of the supported media objects i.e. A/V Control object, video/broadcast object and HTML5 media elements. Otherwise shall be set to true.

A.2.20.2 Extensions for time-shift

The following property is added to the `Configuration` class.

`readonly Boolean timeShiftSynchronized`
Returns a boolean indicating if the terminal is capable of maintaining synchronization between applications and A/V components during time-shift. A definition of synchronization between applications and A/V components can be found in clause 6.2.2.4.

A.2.20.3 Extensions to represent audio description presentation

This class shall be extended with the following additional property.

`readonly Boolean audioDescriptionEnabled`
Shall be set to `false` if audio description is disabled by the terminal, otherwise shall be set to `true`. If set to `false`, applications should not enable audio description using the component selection API of the supported media objects i.e. A/V Control object, video/broadcast object and HTML5 media elements.

A.2.20.4 Extensions for access to network IDs

This class shall be extended with the following additional property:

`readonly Number dtt_network_ids[]`
Returns the ordered list of DVB network_ids from the DTT channels, if any, that are included in the terminal's channel list. If the terminal does not have a DTT receiver or no DTT channels are present in the channel list then the property shall be `undefined`.

A.2.20.5 Extensions for device IDs

The following property is added to the `Configuration` class.

`readonly String deviceId`
Returns a string representing an identifier that is unique for the combination of the receiver and the HTML document origin. The identifier shall use a character set that is restricted to alphanumeric characters and the hyphen. If this identifier is not being made available to the application (see clause 12.1.5 of the present document), then the value of this property shall be the empty string ("").

A.2.21 Void

A.2.22 Modifications to clause 8.4.2

The following modifications shall be applied to clause 8.4.2 of the OIPF DAE specification [1]:

- In the row of the table for the type property, in the columns for MPEG-2 transport streams, the following item shall be extended as shown underlined:

  A value of 0x03 or 0x04 or 0x11 in the `stream_type` field in the PMT -> AUDIO.

A.2.23 AVAudioComponent

In clause 7.16.5.4.1 of the OIPF DAE specification [1], the definition of the `audioChannels` property shall be extended as shown:

Indicates the number of main channels present in this stream (e.g. 2 for stereo, 5 for 5.1, 7 for 7.1). Potentially available low frequency effects channels are not included in this indication.

A.2.24 Modifications to Clause 7.10.1.1 and references to it

In clause 7.10.1.1;

Firstly the description of the `getScheduledRecordings()` method is modified as shown;

`ScheduledRecordingCollection getScheduledRecordings()`	
Description	Returns ~~a subset of all the~~ those recordings that are scheduled but which have not yet started and which were scheduled by an application from the same origin as the caller. ~~The subset SHALL include only scheduled recordings that were scheduled using a service from the same FQDN as the domain of the service that calls the method.~~

Secondly the description of the `remove()` method is modified as shown;

`void remove( ScheduledRecording recording )`		
Description	Remove a recording. ~~For non-privileged applications, recordings SHALL only be removed when they are scheduled but not yet started and the recording was scheduled by the current service.~~ Recordings SHALL only be removed when they are scheduled but not yet started. Additionally for terminals with the attribute `manageRecordings` set in the `<recording>` element of their capabilities set to `"samedomain"`, recordings shall only be removed when the recording was scheduled by applications from the same origin as the caller. As with the `record()` method, only the `programmeID` property of the scheduled recording SHALL be used to identify the scheduled recording to remove where this property is available. The other data contained in the scheduled recording SHALL NOT be used when removing a recording scheduled using methods other than `recordAt()`. For recordings scheduled using `recordAt()`, the data used to identify the recording to remove is implementation dependent. If the `programmeIDType` property has the value `ID_TVA_GROUP_CRID` then the OITF SHALL cancel the recording of the specified group. If an A/V Control object is presenting the indicated recording then the state of the A/V Control object SHALL be automatically changed to 6 (the error state).	
Arguments	recording	The recording to be removed

Thirdly the `getInProgressRecordings()` method is added as shown;

`ScheduledRecordingCollection getInProgressRecordings()`	
Description	Returns those recordings that are currently in progress (i.e. those where recording has started but has not yet completed) and which were scheduled by an application from the same origin as the caller.

In clause 9.3.3, the following two modifications are made as shown;

> The Boolean attribute `manageRecordings` specifies whether or not the OITF supports managing recordings through the JavaScript APIs defined in section 7.10.4 and 7.10.1.
>
> `"samedomain"`: indicates that recordings initiated by applications from the same ~~fully-qualified domain~~ origin may be managed

A.2.25 Modifications to content download descriptor and content access streaming descriptor

The following extension of the Content Access Download Descriptor and Content Access Streaming Descriptor shall be supported.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="urn:hbbtv:CAD:2014"
           xmlns:hbbtv="urn:hbbtv:CAD:2014"
           xmlns:oipf1="urn:oipf:iptv:ContentAccessDownloadDescriptor:2008-1"
           xmlns:oipf2="urn:oipf:iptv:ContentAccessStreamingDescriptor:2008-1"

           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
 <xs:import namespace="urn:oipf:iptv:ContentAccessDownloadDescriptor:2008-1"
            schemaLocation="oipf\iptv-ContentAccessDownloadDescriptor.xsd"/>
 <xs:import namespace="urn:oipf:iptv:ContentAccessStreamingDescriptor:2008-1"
            schemaLocation="oipf\iptv-ContentAccessStreamingDescriptor.xsd"/>
  <xs:complexType name="DownloadContItemType">
    <xs:complexContent>
      <xs:extension base="oipf1:ContItemType">
        <xs:sequence>
          <xs:element name="DownloadableFont" type="hbbtv:DownloadableFontType"
                      minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="CICAMPlayerPreferred" minOccurs="0" maxOccurs="1"/>
</xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="StreamingContItemType">
    <xs:complexContent>
      <xs:extension base="oipf2:ContItemType">
        <xs:sequence>
          <xs:element name="DownloadableFont" type="hbbtv:DownloadableFontType"
                      minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="DownloadableFontType">
    <xs:simpleContent>
      <xs:extension base="xs:anyURI">
        <xs:attribute name="font-family" use="required" type="xs:string"/>
        <xs:attribute name="mime-type" use="required" type="xs:string"/>
        <xs:attribute name="essential" use="optional" type="xs:boolean" default="false"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:schema>
```

The `DownloadableFontType` attributes are defined as follows:

- **font-family:** the terminal shall use the downloadable font if the font used in an EBU-TT-D document matches this font-family.
- **mime-type:** font formats are identified by a mime-type as listed in the DVB DASH specification TS 103 285 [45].
- **essential:** defines whether the downloadable font is essential to render the subtitles.

The text content of the `DownloadableFontType` signals the HTTP download location of the font file. Multiple `<DownloadableFont>` elements may be present to signal different formats of the same font family using the `@font-family` attribute. The terminal shall use the `@mime-type` attribute to choose a supported format.

The `CICAMPlayerPreferred` element indicates if the HbbTV terminal shall give preference when playing back the content to a CICAM player (as defined in clause 11.4.5 and annex K of the present document) if one is present.

A.3 Modifications, extensions and clarifications to volume 5a

A.3.0 General

The following modifications shall be made to the profile of HTML5 [54] and other web standards defined by the OIPF Web Standards TV Profile [i.6].

A.3.1 Additional support for TextTracks and Cues

The following elements and properties shall be supported by terminals in addition to those required by clause A.1.3.1 of the OIPF Web Standards TV Profile [i.6]:

- The `HTMLTrackElement` interface and the `<track>` element in an HTML document.
- For the `HTMLMediaElement.textTracks` property, support for out-of-band tracks is required in addition to support for in-band tracks which is required by that specification.
- From the `TextTrack` class:
  - The `id`, `inBandMetadataTrackDispatchType`, `cues` and `activeCues` properties.
  - The `oncuechange` event.
- `TextTrackCueList`.
- `TextTrackCue` shall be supported as the parent for `DataCue`.
- `DataCue` (see HTML 5.1 [51]) shall be supported by terminals for DASH events as defined in clause 9.3.2.2 of the present document.

Specifically there is no requirement to support any of the following:

- Instances of `TextTrackCue` which are not also instances of `DataCue`.
- Instances of `DataCue` (and hence also `TextTrackCue`) except for those created by the terminal to report DASH events to applications as defined in clause 9.3.2.2 of the present document.
- Instances of `TextTrack` other than those required to be created by clause A.2.12.1 of the present document or created using the HTML track element.

Instances of `TextTrackCue` (or any sub-class) and the methods `addCue()` and `removeCue()` should not be supported for TTML in order to avoid unpredictable interactions with the HbbTV® terminal's internal TTML decoder. For example cues being rendered in duplicate.

A.3.2 Additional support for getStartDate in HTML5 media elements

The `getStartDate()` method of HTML5 media elements shall be supported by terminals in addition to the requirements of clause A.1.3.1 of the OIPF Web Standards TV Profile [i.6]. The method shall be implemented for MPEG DASH content as defined in clause 9.4.2 of the present document.

A.3.3 Event model

Clause 6.2 of the OIPF Web Standards TV Profile [i.6] does not apply in the present document. Instead "keydown", "keypress" and "keyup" events shall be supported as defined by annex B of the Open IPTV Forum DAE specification release 1 [53] under the heading "Add keypress events to Requirement 5.4.1.a in the following way".

A.3.4 Resize event

In clause A.1.3.2 "Media Element Events Support" of the Web Standards TV Profile [i.6], the resize event defined in HTML5 [54] shall also be supported.

A.3.5 HTML5 recommendation

In the Web Standards TV Profile [i.6], the reference to HTML5 shall be changed from:

Robin Berjon; Steve Faulkner; Travis Leithead; Erika Doyle Navara; Edward O'Connor; Silvia Pfeiffer. HTML5. 6 August 2013. W3C Candidate Recommendation. URL: http://www.w3.org/TR/2013/CR-html5-20130806/.

To:

Ian Hickson; Robin Berjon; Steve Faulkner; Travis Leithead; Erika Doyle Navara; Edward O'Connor; Silvia Pfeiffer. HTML5: A vocabulary and associated APIs for HTML and XHTML; W3C Recommendation 28 October 2014.

NOTE: The following should be noted as a consequence of this change:

The `scoped` attribute of the `HTMLStyleElement` which is made optional by [i.6] is dropped from HTML5 by this change. As a consequence, this element becomes fully supported.

The `seamless` attribute of the `<iframe>` element which is made optional by [i.6] is dropped from HTML5 by this change. However the `sandbox` and `srcdoc` attributes also made optional by [i.6] remain in HTML after this change hence this element remains partially supported.

The `<command>`, `<details>` and `<summary>` elements which are made optional by [i.6] are dropped from HTML5 by this change.

The `<menu>` element is not required to be supported, not even partially as required by [i.6].

A.3.6 Support for volume controls

The requirements for the following elements and properties are modified with respect to the OIPF Web Standards TV Profile [i.6]:

- There is no requirement to support the `volume` attribute of the `HTMLMediaElement` on terminals that do not support multiple audio decoders.

A.3.7 Support for multiple audio tracks

The requirements for the following elements and properties are modified with respect to the OIPF Web Standards TV Profile [i.6]:

- There is no requirement to support multiple simultaneously enabled audio tracks on an HTML5 media element. Enabling a new audio track shall automatically disable the previous one on terminals that are unable to decode multiple audio tracks and mix them.

A.3.8 Fonts

Clause 5.3 of the OIPF Web Standards TV Profile [i.6] requires support for the Web Open Font Format (WOFF).

Terminals are strongly recommended to employ measures to sanitise downloaded font files to reduce risks arising from font parser vulnerabilities. Note: This may be achieved by use of a font sanitiser [i.13].

Content providers shall ensure that all font files used by HbbTV Applications strictly conform to the relevant specifications. Non-compliant fonts are likely to be rejected by terminals that employ a font sanitiser.

A.3.9 Support for high resolution graphics

In addition to the properties required by clause 6.3 of the Web Standards TV Profile [i.6], terminals shall support the `devicePixelRatio` property of the `Window` interface. The value returned shall reflect the physical pixel resolution at which the HbbTV application is rendered and displayed.

NOTE: The physical pixel resolution may be higher than the logical resolution implied by the HbbTV application's co-ordinate system as defined in 10.2.1.

Table A.5 shows example values for `Window.devicePixelRatio` for commonly used rendering resolutions.

Table A.5: Window.devicePixelRatio values

Rendering resolution	Value
1 280 x 720	1.0
1 920 x 1 080	1.5
3 840 x 2 160	3.0

If the terminal uses a graphics plane resolution other than 1 280 x 720 then:

- The `srcset` and `sizes` attributes of the `img` element shall be supported as defined by HTML 5.1 [51].
- When an `img` element is used to present an image at a particular target size such that for each axis, the size of the image asset in pixels is less than or equal to the number of device pixels present within the rendering region for the `img` element, the image shall be rendered without loss of resolution.

  For example, on a terminal with a rendering resolution of 1 920 x 1 080, a PNG image of size 150x150 being presented in an `img` element with width and height 100 CSS pixels shall be presented without loss of resolution.
- In general, the algorithm for selecting the most appropriate image source from a `srcset` is outside the scope of the present document. However, when an `img` element has a valid `srcset` attribute that includes an entry with a pixel density descriptor that matches the device pixel ratio, the image source corresponding to that entry shall be selected. Similarly, if the `sizes` attribute is present and an image listed in the `srcset` attribute has an effective pixel density that matches the device pixel ratio, that image shall be selected.

  For example, when presented with an `img` element with a `srcset` attribute of "`low.png 1x, medium.png 1.5x, high.png 3x`", a terminal with a rendering resolution of 1 920 x 1 080 shall select "`medium.png`" as the image source and a terminal with a rendering resolution of 3 840 x 2 160 shall select "`high.png`".

  Similarly, when presented with an img element with a `srcset` attribute of "`low.png 640w, medium.png 960w, high.png 1920w`" a `sizes` attribute of "`640px`" and a `width` attribute of "`640`", a terminal with a rendering resolution of 1 920 x 1 080 shall select "`medium.png`" as the image source and a terminal with a rendering resolution of 3 840 x 2 160 shall select "`high.png`".

A.3.10 Web Audio API

The following interfaces from the Web Audio API [65] shall be supported with the properties and methods listed for each:

- AudioNode: connect,disconnect, context, numberOfInputs, numberOfOutputs, channelCount, channelCountMode, channelInterpretation
- AudioBufferSourceNode: buffer, start, stop, onended
- AudioContext: createBuffer, createBufferSource, destination, sampleRate, currentTime, decodeAudioData
- AudioBuffer: length, sampleRate, duration, numberOfChannels
- AudioDestinationNode: maxChannelCount

For the decodeAudioData method, the requirement that "Audio file data can be in any of the formats supported by the audio or video elements" does not apply. See Table 11 for the required codec support.

A.3.11 Encrypted media extensions

The W3C encrypted media extensions API [66] shall be supported for encrypted content delivered via broadband as defined in clause B.3 of the present document. Use of this API with a DRM system is outside the scope of the present document.

A.3.12 CSS

HbbTV applications shall use CSS without vendor prefixes. Using prefixed CSS is discouraged as this may lead to interoperability issues.

A.3.13 Mixed content

Application developers should be aware that HbbTV terminals may implement the W3C Mixed Content specification [i.18] (subject to the requirements in this clause) and should write applications such that they work correctly on such terminals.

An HbbTV terminal that implements the Mixed Content specification [i.18] shall not consider video or audio loaded via `<video>` or `<audio>` elements or the A/V control object as blockable content for the purposes of protecting against mixed content.

Regardless of how an HbbTV application was delivered (HTTP, HTTP over TLS ("https:"), object carousel), terminals shall permit them to successfully make WebSocket connections to WebSocket endpoints that have been requested and returned to the application via one of the following APIs from clause 14.5.2:

- `getApp2AppLocalBaseURL`
- `discoverTerminals` (if supported)

NOTE 1: This requirement conflicts with the requirement in the Mixed Content specification concerning WebSocket but offers greater security than if HbbTV applications using application to application communication were required to be delivered by plain HTTP (i.e. not using TLS), particularly in the context of a TV device that does not typically present origin or page security information to the viewer.

NOTE 2: The present document is intentionally silent about how HbbTV terminals that implement the Mixed Content specification also meet the above requirement. A number of possibilities exist depending on the security requirements that the implementer needs to address and how much freedom and/or expertise they have to modify the browser. As one example, the URLs that the above APIs provide to the HbbTV application for the WebSocket endpoints could be something other than 'ws' or 'wss'.

NOTE 3: Using a private certificate authority to issue certificates for the WebSocket server (and hence permit secure WebSockets to be used) is strongly discouraged due to the security risks to other TLS-secured communications that are likely to result. Such an approach would not work properly with remote WebSockets endpoints as some clients could not be configured to accept private certificate authorities.

A.4 Modifications, extensions and clarifications to volume 7

A.4.1 Processing of the CI parental_control_info message

Section 4.2.3.4.1.1.5 shall be modified as follows:

> When the parental_control_info message is received and a DAE application is launched, the OITF SHALL issue the relevant event to the DAE application:
>
> - `onParentalRatingChange` event, if the parental rating system specified by the `oipf_rating_type` is supported by the OITF.

- `onParentalRatingError` event, if the parental rating system specified by the `oipf_rating_type` is not supported by the OITF.

NOTE: When processing a parental_control_info message, an OITF supporting (or not) for a parental rating system is only used to determine which event is issued to a DAE application (as above) and to set the attributes of the event (as below) for supported parental rating systems. Parental rating thresholds and PIN codes set in the terminal are not used in this process and the terminal does not generate an UI.

The prototype of the `onParentalRatingChange` and `onParentalRatingError` events defined in [DAE] are recalled here:

```
function onParentalRatingChange( String contentID, ParentalRating rating, String DRMSystemID,
Boolean blocked )
function onParentalRatingError( String contentID, ParentalRating rating, String DRMSystemID)
```

Annex B (normative): Support for protected content delivered via broadband

B.1 Introduction

When content protection is being used, the type of content protection in use shall be signalled:

- as defined in clause 9.3.10 of the OIPF DAE specification [1] and in Table 9 ("DRMControlInformation Type Semantics") of the OIPF Metadata specification [18];
- using DVB-CA identifier codepoints (CA_System_ID) allocated as usual by the DVB Project and found in TS 101 162 [19] for the DRMSystemID.

Some issues that need to be considered when defining how a particular content protection technology is integrated with implementations of the present document are described in annex F.

B.2 Common Encryption for ISOBMFF

NOTE: The requirements formerly found in this clause are replaced by those in clause 8 of the DVB DASH profile TS 103 285 [45].

B.3 Clear key encryption

The “Clear Key” key system defined in clause 9.1 of the encrypted media extensions API [66] shall be supported for content delivered by MPEG DASH (see clause E of the present document). The requirements in clause B.1 of the present document do not apply to this key system. The initialization data type '`cenc`' shall be supported - see clause 8.3 of EME [66].

Support for the “Clear Key” key system does not imply support for the “DRM feature” and the requirements in the present document that apply if that feature is supported shall not apply to “Clear Key”.

Receivers shall not allow content delivered in this way to be:

- Redistributed through any local or remote network without encryption.
- Presented through any HDMI output without HDCP being enabled.

NOTE: Export of content may be subject to further constraints imposed by other contractual or legal agreements.

Annex C (informative): Support for analogue broadcasting networks

C.1 Scope

The main target of the HbbTV® specification is to combine services delivered via a DVB compliant broadcast network and a broadband connection to the Internet. Many of the conceptual and technical aspects of HbbTV, however, are also applicable to a combination of an analogue Broadcast network and a broadband Internet connection. Analogue TV distribution may for some years still be of relevance for some markets.

If a terminal includes an analogue front end, the HbbTV® concept may be applied to analogue channels as described in this annex. If the HbbTV® concept is not applied to analogue channels then they would be treated in the same way as DVB channels without an AIT.

C.2 AIT retrieval and monitoring

As the AIT cannot be provided within the analogue broadcast channel, it has to be retrieved via the Internet connection. When tuning to an analogue service the hybrid terminal can send an http request to a server hosting AIT information as following.

```
http://[AIT_server]/service?CNI=xxx
http://[AIT_server]/service?name=xxx
```

This request will return the AIT of the corresponding service encoded in XML format as defined in TS 102 809 [3]. The AIT is contained in a single application discovery record.

The IP address or the base URL of the AIT server may be market or manufacturer specific. It could be part of the default settings of the terminal and may allow for changes by the user.

For the identification of the service the CNI code as registered in TS 101 231 [i.3] should be used. As an alternative the name of the service may be used.

AIT monitoring while being tuned to a specific service can be done by repeating the http requests defined above. The xml document that contains the AIT carries a version attribute within the `<ServiceDiscovery>` element. If present the version attribute is used in the request as follows:

```
http://[AIT_server]/service?CNI=xxx&version=YY
http://[AIT_server]/service?name=xxx&version=YY
```

where `YY` are two hexadecimal digits. If the recent version on the server is the same as in the request the server returns the HTTP status code 204 with no message body.

The repetition rate should not be more frequent than once per 30 seconds.

C.3 Tuning to a new channel

The video/broadcast embedded object defined in the OIPF DAE specification [1] can be used to determine available analogue broadcast services and to tune between them as described in this clause.

An analogue broadcast service is represented by a channel object with an idType of `ID_ANALOG` including the properties `cni` and/or `name`. The `cni` property contains the CNI of the service when it is available in the broadcast signal. The name property is available when the CNI is not broadcast. For CNI and name see clause C.2.

The channel line-up of the HbbTV® terminal is available to the application in order to be able to retrieve channel objects for a CNI or name.

The `currentChannel` property on the video/broadcast object and the `ApplicationPrivateData.currentChannel` property returns the channel object for the analogue service currently presented.

C.4 Other aspects

EIT access, application transport with DSM-CC, stream events, etc. are not available on analogue channels. Method calls related to these features cause exceptions with a message "not supported". Properties related to these features have the value `undefined`.

Annex D (informative): Server root certificate selection policy

D.1 Introduction

This informative annex describes the policy that is adopted for the selection of root certificates for inclusion in terminals compliant with the present document. A list of such certificates is published at http://www.hbbtv.org/spec/certificates.html.

D.2 Background

There are over 150 root certificates in web browsers at the time of publication.

- This list changes frequently over time.
- The larger the list of root certificates the more likely it is to change.

The security of TLS against man-in-the-middle attacks is dependent on the weakest root certificate trusted by a terminal.

The security of various key lengths changes with time as computing power increases. Specifically 1 024 bit RSA keys are no longer recommended for use.

Service providers need to know which root certificates are trusted by terminals to achieve interoperability. Service providers are often not in control of the servers delivering their content (e.g. delivery via a CDN).

Service providers may also wish to make use of third party web services that are not under their control.

Maintaining an independent list of root certificates that are validated requires significant resources.

D.3 Policy

The Mozilla list of approved root certificates has been selected as the authoritative source for the mandatory and optional list of root certificates for inclusion in terminals compliant with the present document. This was chosen because:

- The approved root certificate list is publicly available.
- The process for inclusion in the list is open.
- Anyone can take part in the acceptance process.
- The acceptance process itself happens in public.
- Metadata is provided to differentiate root certificates for web server authentication, e-mail and code signing.
- The procedure for requesting a root certificate for inclusion in the list requires a test website be provided which uses that certificate.

The Mozilla list of approved root certificates is published on their website at http://www.mozilla.org/projects/security/certs/. Each certificate marked as approved for web server authentication is automatically an optional root certificate as specified in clause 11.2.

The present document will rely upon the Mozilla list for verifying the trustworthiness of Certificate Authorities.

A list of root certificates that are mandatory will be maintained which will be a subset of the certificates specified above.

- The list will be updated periodically.

- The list will only include certificates that use algorithms mandated by clause 11.2.4.
- The mandatory list of certificates will be determined based on the requirements of service providers and the Certificate Authorities that are in widespread use.
- The list will be compiled relying upon published statistics to determine how widespread a Certificate Authority is.
- Certificate Authorities may be excluded from the mandatory list if they impose requirements that are deemed unreasonable.
- A revision history of changes to the mandatory list will be maintained and published.

This policy is subject to change.

Annex E (normative): Profiles of MPEG DASH

E.1 Introduction (informative)

This annex defines some minor additional requirements and constraints to the DVB DASH profile TS 103 285 [45]. Most of the text in earlier versions of this annex is replaced by text in that document. When those additional requirements and constraints are included, the DVB DASH profile is believed to support all content compatible with this annex in earlier versions of the present document.

Unlike the previous version of this annex, this version also supports adaptive delivery of radio services.

E.2 Requirements relating to the MPD

E.2.1 Profile definition

The MPD shall indicate either or both of the following profiles:

- the profile of DASH defined by DVB in TS 103 285 [45] ("`urn:dvb:dash:profile:dvb-dash:2014`");
- "`urn:hbbtv:dash:profile:isoff-live:2012`" as used in previous versions of the present document.

Terminals may raise an error to the application when a referenced MPD does not contain either of these profiles in the `@profiles` attribute. Terminals shall be able to play the content described by the profile-specific MPD (as defined in clause 8.1 of DASH ISO/IEC 23009-1 [29]) (but not necessarily other Adaptation Sets or Representations in the MPD discarded as part of the process of deriving the profile-specific MPD).

The following clauses define the additional restrictions and requirements on an MPD identified as conforming to the DVB profile, as well as requirements on terminals when playing such content. Additionally:

- The profile specific MPD shall include at least one Adaptation Set encoded using the audio or video codecs defined in clause 7.3.1 of the present document. Adaptation Sets and Representations in non-supported codecs shall be ignored.

E.2.2 Numerical requirements

NOTE: The numerical constraints formerly in this clause can now be found in clause 4.5 of the DVB DASH profile TS 103 285 [45].

The behaviour of a terminal is undefined for MPDs that do not comply with the requirements in that clause.

E.2.3 Metadata requirements

NOTE: The requirements formerly in this clause can now be found in clause 4.4 and 6.1.2 of the DVB DASH profile TS 103 285 [45].

E.2.4 Role Related requirements

NOTE: The requirements formerly in this clause can now be found in clause 6.1.2 of the DVB DASH profile TS 103 285 [45].

E.2.5 Audio Channel Configuration requirements

For E-AC-3 the Audio Channel Configuration shall use either the "`tag:dolby.com,2014:dash:audio_channel_configuration:2011`" (as defined in the DVB DASH profile TS 103 285 [45]) or the legacy "`urn:dolby:dash:audio_channel_configuration:2011`" schemeURI.

NOTE: The other requirements formerly in this clause can now be found in clauses 6.1.1, 6.2 and 6.3 of the DVB DASH profile TS 103 285 [45].

E.2.6 Content protection signalling

NOTE: The requirements formerly in this clause can now be found in clause 8.4 of the DVB DASH profile TS 103 285 [45].

E.3 Restrictions on content

E.3.1 Restrictions on file format

E.3.1.1 ISO Base Media File Format

NOTE: The requirements formerly in this clause can now be found in clauses 4.3 and 10.2 of the DVB DASH profile TS 103 285 [45].

E.3.2 Restrictions on adaptation sets

NOTE: The requirements formerly in this clause can now be found in clauses 4.2.4, 4.3, 4.5 and 5.1.2 of the DVB DASH profile TS 103 285 [45].

E.4 Requirements on terminals

E.4.1 DASH profile support

Terminals shall support the DVB DASH profile TS 103 285 [45] as modified in the present document. MPDs that identify themselves with the profile "`urn:hbbtv:dash:profile:isoff-live:2012`" and not "`urn:dvb:dash:profile:dvb-dash:2014`" shall be supported as if they had indicated "`urn:dvb:dash:profile:dvb-dash:2014`" and "`urn:dvb:dash:profile:dvb-dash:isoff-ext-live:2014`".

Other profiles (e.g. the DASH-IF DASH-AVC/264 main interoperability point (see [i.19])) may be supported.

The following rules apply for MPDs that do not list either or both of "`urn:hbbtv:dash:profile:isoff-live:2012`" or "`urn:dvb:dash:profile:dvb-dash:2014`" in their MPD `@profiles` atttribute;

- MPDs specified with the "`urn:mpeg:dash:profile:isoff-live:2011`" profile should be supported.
- MPDs specified with the "`urn:mpeg:dash:profile:isoff-on-demand:2011`" profile should not be supported unless the HbbTV terminal supports the parts of that profile that are not required by the DVB-DASH profile [45], such as multiple ‘sidx’ boxes.
- MPDs specified with the URNs defined for the interoperability points defined in the DASH-IF guidelines [i. 19] shall be rejected by HbbTV terminals that do not support the specified inter-operability point.
- MPDs specified with profiles beginning "`urn:hbbtv:dash:profile`" shall be rejected unless that profile is defined in a later version of the present document and the HbbTV terminal supports the specified profile.
- MPDs specified with profiles beginning "`urn:dvb:dash:profile`” shall be rejected unless that profile is defined in a later version of the DVB-DASH specification than V1.1.1.[45] and the HbbTV terminal supports the specified profile.
- If an MPD specifies multiple profiles (but neither of those required to be supported by the present document) where some of them are required to be rejected by the rules in this clause and others are not required to be rejected by those same rules then the MPD is not required to be rejected.

The following rules apply for Adaptation Sets and/or Representations that are not indicated as conforming to at least one of the "`urn:dvb:dash:profile:dvb-dash:isoff-ext-live:2014`" or "`urn:dvb:dash:profile:dvb-dash:isoff-ext-on-demand:2014`" or "`urn:hbbtv:dash:profile:isoff-live:2012`" profiles:

- Adaptation Sets or Representations indicated as being compliant with "urn:mpeg:dash:profile:isoff-on-demand:2011" should be ignored unless the HbbTV terminal supports the parts of that profile that are not required by the DVB-DASH profile [45], such as multiple 'sidx' boxes.
- Adaptation Sets or Representations indicated as being compliant with "urn:mpeg:dash:profile:isoff-live:2011" should not be ignored unless there are other reasons to do so (e.g. non-supported codec or @role).
- Adaptation Sets or Representations indicated as being compliant with one or more of the interoperability points in the DASH-IF interoperability guidelines [i.19] shall be ignored by HbbTV terminals that do not support that inter-operability point.
- Adaptation Sets or Representations indicated as being compliant with profiles beginning "urn:hbbtv:dash:profile" shall be ignored unless the indicated profile is defined in a later version of the present document and the HbbTV terminal supports that profile.
- Adaptation Sets or Representations indicated as being compliant with profiles beginning "urn:dvb:dash:profile" shall be ignored unless the indicated profile is defined in a later version of the DVB-DASH specification than V1.1.1.[45] and the HbbTV terminal supports the indicated profile.
- Where the MPD @profiles attribute includes either or both of "urn:hbbtv:dash:profile:isoff-live:2012" or "urn:dvb:dash:profile:dvb-dash:2014" as well as some other profile, AdaptationSets and Representations not inferred to have a @profiles attribute that includes one of "urn:dvb:dash:profile:dvb-dash:isoff-ext-live:2014", "urn:dvb:dash:profile:dvb-dash:isoff-ext-on-demand:2014" or "urn:hbbtv:dash:profile:isoff-live:2012" shall be ignored by HbbTV terminals that support only the DVB DASH profile [45].
- Adaptation Sets or Representations indicated as being compliant with multiple profiles (but none of those required to be supported by the present document) where some of the profiles are required to be ignored by the rules in this clause and others are not required to be ignored by those same rules are not required to be ignored.

E.4.2 Transitions between representations

E.4.2.1 Video tracks

NOTE: The requirements formerly in this clause can now be found in clauses 10.3 and 10.4 of the DVB DASH profile TS 103 285 [45].

In addition to the set of resolutions in Table 18 in the DVB DASH profile TS 103 285 [45], 720 x 576i shall also be supported.

E.4.2.2 Audio tracks

NOTE: The requirements formerly in this clause can now be found in clause 10.4 of the DVB DASH profile TS 103 285 [45].

E.4.3 Buffering

NOTE: The requirements formerly in this clause can now be found in clause 10.7 of the DVB DASH profile TS 103 285 [45] and clause 10.2.3.2 of the present document.

E.4.4 ISO File Format support

NOTE: The main requirements relating to this subject are now found in clause 10.2 of the DVB DASH profile TS 103 285 [45].

Additionally terminals shall support both the '`avc1`' and '`avc3`' sample entry types for H.264 content that are referred to in clause 5.1.2 of that profile.

E.4.5 MPD Anchors

Terminals shall support MPD Anchors using the `'t'` key of the URI fragment part as defined in clause C.4 of the MPEG DASH specification ISO/IEC 23009-1 [29] as profiled in clause 10.9.2 of the DVB DASH profile TS 103 285 [45]. Support for other MPD Anchor keys is not required.

Annex F (informative): DRM Integration

F.1 Introduction

This annex identifies issues which need to be considered and in most cases documented when defining how a DRM system is to be integrated with HbbTV®. It is expected that solutions to these issues would form the basis of the document defining the technical integration between HbbTV® and that DRM system and subsequently a test specification and test suite.

F.2 General issues

Some informative text is needed identifying how the key aspects of the DRM technology map on to the mechanisms and local interfaces showing in annex D of the OIPF DAE specification [1].

A DRM System ID for the DRM system needs to be registered in as described in the OIPF DAE specification [1], clause 9.3.10.

If the DRM agent can generate user interfaces on the terminal then the interaction between these and the HbbTV® system needs to be defined. This is particularly critical if these user interfaces are rendered using the same browser as is used for HbbTV® applications. (See the OIPF DAE specification [1], clause 5.1.1.6).

Which combinations of protocols and codecs are required to be supported with the DRM technology need to be defined. These need to be in the format of the video profile capability strings indicating as defined in the OIPF DAE specification [1], clause 9.3.11.

F.3 DRM Agent API

In the `sendDRMMessage` method (as defined in the OIPF DAE specification [1], clause 7.6.1.2), it needs to be defined which values of the `msgType` parameter are valid and what the contents of the `msg` parameter are for each message type.

In the `onDRMMessageResult` function (as defined in the OIPF DAE specification [1], clause 7.6.1.1), the valid values for the `resultMsg` parameter should be defined if they are intended to be parsed by an HbbTV® application. Additionally it needs to be defined which conditions in the DRM system trigger which `resultCode` values and any implications on the value of the `resultMsg`.

F.4 Content via the A/V Control object

If DRM is used to protect content presented via the A/V Control object then the following need to be specified:

1) Whether the content access streaming descriptor is needed to provide information for the DRM system. If so then which of the fields are used, under what circumstances and what the requirements are on their contents need to be defined. If not then the mechanism by which DRM information is obtained needs to be defined.

2) Whether the DRM system can enforce parental access control and trigger an `onParentalRatingChange` event (as defined in the OIPF DAE specification [1], clause 7.14.5). If this event can be triggered then how the value of the `contentID parameter` is obtained needs to be specified. The same applies for `onParentalRatingError` event.

3) The conditions when the `onDRMRightsError` event is generated (as defined in the OIPF DAE specification [1], clause 7.14.6). If it is generated, the values to be used for the `contentID` and the `rightsIssuerURL` parameters need to be defined.

F.5 Content via the HTML5 media element

If DRM is used to protect content presented via the HTML5 media element, then the following need to be specified:

1) Whether the content access streaming descriptor is needed to provide information for the DRM system. If so then which of the fields are used, under what circumstances and what the requirements are on their contents need to be defined. If not then the mechanism by which DRM information is obtained needs to be defined.

2) How detection and handling of errors need to be defined - see also clause 9.6.7.

Annex G (informative): Implementer guidelines for media synchronization

G.1 General

Annex G provides implementer guidelines for media synchronization. It focusses on the broadcaster perspective. Whereas the present document defines only the terminal behaviour, provisions by broadcaster are needed to make media synchronization actually work.

Clause G.2 shows how a broadcaster could manage delay throughout distribution network in order to prevent buffer overflow or underrun at the terminal or Companion Screen.

Clause G.3 shows how a broadcaster could manage multiple content timelines and provide correct correlation timestamps for cases where the distribution network make changes to the timeline (e.g. changes to PTS).

More implementation guidelines for broadcasters are provided in annex B of TS 103 268-2 [47].

G.2 Managing delay throughout distribution network

There are several reasons why a broadcaster may want to manage and equalise delays throughout the distribution network(s):

- DVB broadcast streams have typically much lower latency than OTT streams.
- Delays are different in different network segments, e.g. due to transcoding.
- Media-stream buffer capacity is limited in HbbTV® terminal.

Especially for live broadcasts with live companion streams, it is important that media streams arrive at similar times such that there are no buffer overflows or underflows at the user side. Equalizing delays between head-ends can also be beneficial to social TV use cases (out of scope for the present document), where friends or groups of people communicate with each other while watching the same content at different locations, a.k.a. "watching apart together".

Figure sketches an architecture to achieve the required delay management and equalisation.

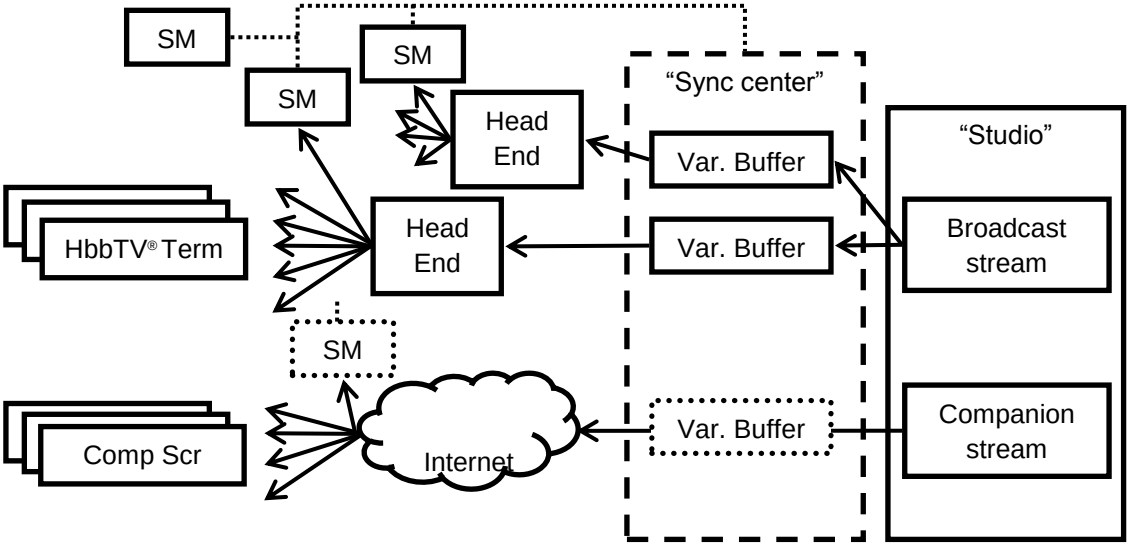


Figure G.1: Architecture for delay management and equalisation

The architecture has a broadcaster studio that provides broadcast and companion streams. The broadcaster is assumed to have a "synchronization center" where synchronization is managed. Stream monitors (SM) are placed at strategic points in the distribution networks to monitor the playout timing of the different network segments, typically at a head end or at a special TV device. The reports from the stream monitors are used at the synchronization center to control variable-delay buffers per network segment and per channel, resulting in a coarse delay equalisation of the different streams. The fine synchronization will happen in/between the HbbTV® terminal and Companion Screen(s) in the home.

G.3 Managing multiple content timelines

The existence of multiple timelines will be a fact of life for a broadcaster, until all its distribution networks support immutable timelines like MPEG TEMI. Head ends of distribution networks typically re-multiplex, transcode and even re-originate broadcast streams. In some head-ends, the broadcast timeline (PCR/PTS) may remain unchanged whereas in other head ends, the broadcast timeline may be stripped and a new broadcast timeline is created. This means at least an unknown offset between PCR/PTS values of the original stream and the new stream(s). Also, there may be subtle variations between the tick rates of the original-stream PCR clock and the new-stream(s) PCR clock(s). The broadcaster will need to handle the situation of having different PCR/PTS in different distribution-network segments.

Figure sketches an architecture to manage multiple content timelines.

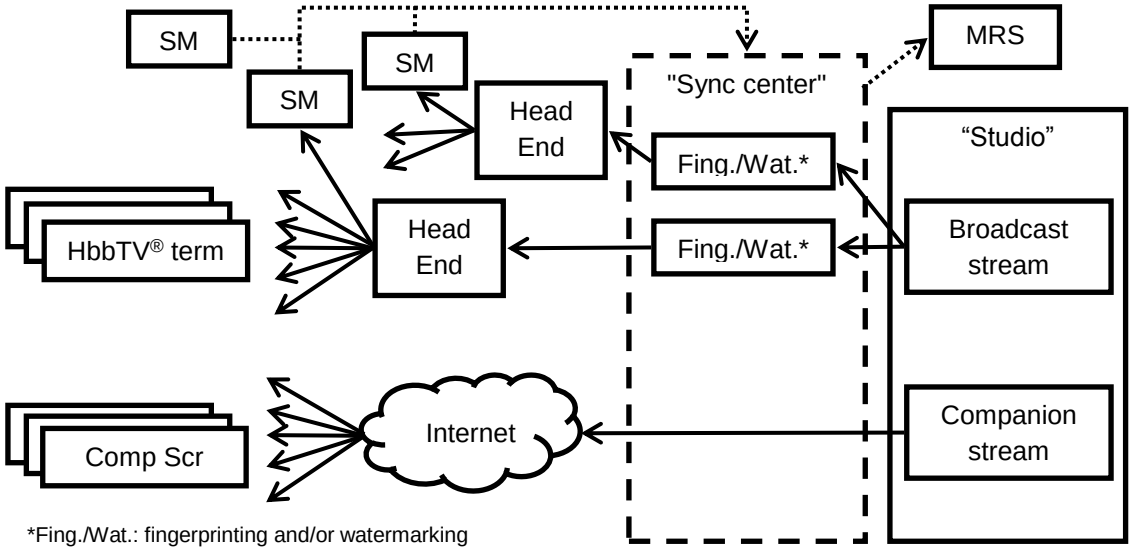


Figure G.2: Architecture for managing multiple content timelines

The architecture has a broadcaster studio that provides broadcast and companion streams. The broadcaster is assumed to have a "synchronization center" where synchronization is managed. Stream monitors (SM) are placed at strategic points in the distribution networks to monitor the relationship between the play-out timing and PCR/PTS, typically at a head end or at a special TV device. The broadcasters may fingerprint and/or watermark the broadcast content such that the stream monitors can correlated the measured timeline (e.g. PCR/PTS) values with a specific point in the content, identified by a fingerprint or watermark. The result is passed to the material resolution server (MRS), such that the MRS can provide HbbTV® terminals material information (MI) expressed in the appropriate broadcast timeline.

G.4 Synchronization with no buffer in the HbbTV® terminal

G.4.0 General

This clause informatively describes the operation of the HbbTV® terminal and the Companion Screen application in the case that the HbbTV® terminal has no buffering capability for broadcast content.

G.4.1 Inter-device media synchronization with the HbbTV® terminal as master with no buffer

Figure G.3 shows the operation of the HbbTV® terminal and the Companion Screen application in the case that the terminal has no buffering capability and is the master device for synchronization.

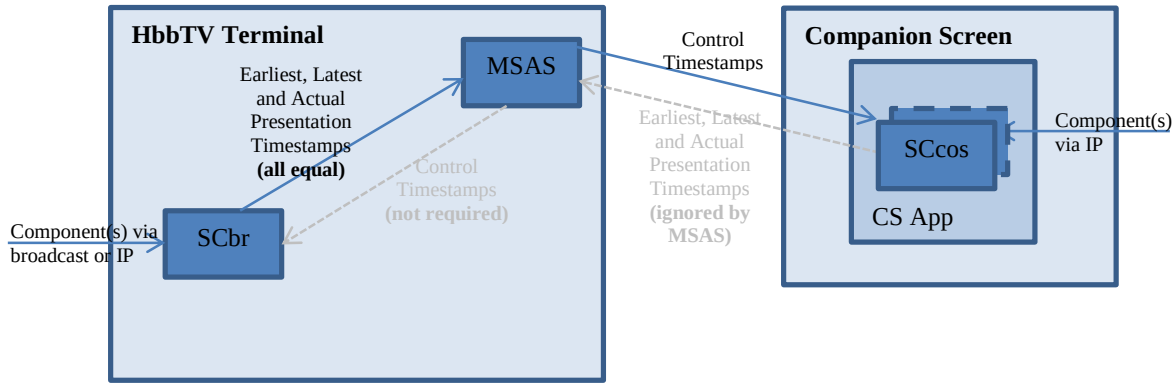


Figure G.3: Architecture for managing multiple content timelines

The SCbr receives and displays one or more components of the service on the HbbTV® terminal (for example, video via broadcast or IP). It provides Earliest, Latest and Actual Presentation Timestamps to the MSAS. For broadcast content, it is not possible to adjust the timing of presentation as there is no buffer. For IP content, there is no need for the HbbTV® terminal to adjust the timing of presentation as the HbbTV® terminal is master. Therefore, Earliest, Latest and Actual Presentation Timestamps are all equal. There is no need for the MSAS to send Control Timestamps to the SCbr as no adjustment is possible, although it is not prevented from doing so (the interface is shown in grey in figure 40).

The MSAS generates Control Timestamps and sends them to the SCcos on the Companion Screen, which is receiving one or more components to be synchronised (for example, an alternative audio track). The SCcos will attempt to render these component(s) according to the Control Timestamps. In case this is not possible, the CSA is responsible for deciding whether to continue rendering the component(s) and/or making any necessary indication to the user.

The SCcos may send Earliest/Latest/Actual Presentation Timestamps to the MSAS, but it is not obliged to do so. As it is not possible to adjust the playout of the component on the HbbTV® terminal, if timestamps are sent by the SCcos, the MSAS can ignore them (the interface is shown in grey in figure 40).

G.4.2 Multi-stream (Intra-device) media synchronization with no buffer for broadcast within the HbbTV® terminal

Figure G.4 shows the operation of multi-stream (intra-device) synchronization within the HbbTV® terminal in the case that the terminal has no buffering capability for broadcast content.

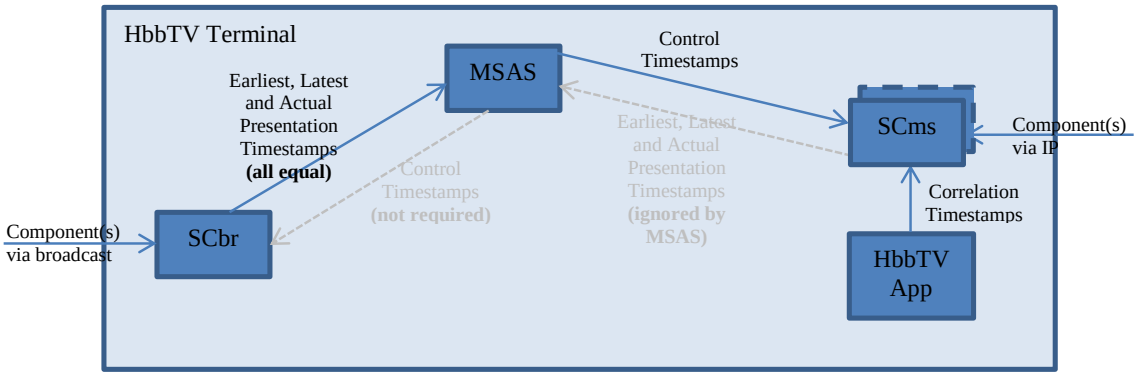


Figure G.4: Multi-stream Media Synchronization with no buffer for broadcast within the HbbTV® terminal

The SCbr receives and displays one or more components of the service from broadcast. It provides Earliest, Latest and Actual Presentation Timestamps to the MSAS. As there is no buffer, it is not possible to adjust the timing of broadcast presentation. Therefore, Earliest, Latest and Actual Presentation Timestamps are all equal. There is no need for the MSAS to send Control Timestamps to the SCbr as no adjustment is possible, although it is not prevented from doing so (the interface is shown in grey in figure G.4).

The MSAS generates Control Timestamps and sends them to the SCms, which is receiving further component(s) to be synchronised. This SC will attempt to render these component(s) according to the Control Timestamps. In case this is not possible, this is reported to the HbbTV® app. Correlation Timestamps are delivered to the SCms from the HbbTV® App. These are used by the SCms to map between the Control Timestamps received from the MSAS and the timeline of the media components that it is receiving.

The SCms will send Earliest, Latest and Actual Presentation Timestamps to the MSAS. However, as it is not possible to adjust the playout of the broadcast component, the MSAS can ignore these (the interface is shown in grey in figure G.4).

Annex H (normative): HbbTV® File Delivery Protocol (FDP)

H.1 High-level principles of FDP (informative)

The broadcast of a file using FDP is based on three categories of messages: an Initialization Message, Data Messages, and a Termination Message.

These messages are carried according to the Data Piping model of the DVB Data Broadcasting specification (EN 301 192 [38]). The present document describes how the messages are encapsulated directly into the payload of the MPEG-2 Transport Stream Packets.

Before being broadcast, the file is divided into segments (the File Segments), each of which is carried in a Data Message.

The Initialization Message is sent before the first Data Message. It provides information regarding the file, which is necessary for the terminal to initialize its reception (file size, segment size, etc.). This is followed by the Data Messages containing the File Segments. Finally, the Termination Message is sent to indicate to the Terminal that this instance of the broadcasting of this file has ended.

The following clauses give details on the usage rules of FDP Messages and their syntax.

Each of these Messages ends with a CRC, allowing the terminal to check the integrity of the received message.

The FDP protocol makes provision for indicating for each file a URL where the terminal may retrieve via broadband the File Segments which have not been successfully received via broadcast (see clause H.3.4 for details).

The current document does not specify any mechanism for error or erasure correction (FEC). However, the FDP protocol has been designed in such a way that it can be extended in the future with one (or more) error/erasure correction scheme, whilst keeping compatible with terminals not implementing it.

H.2 Encapsulation and signalling

H.2.1 DVB signalling

The broadcasting of the files is done on DVB Data Broadcast services using Data Piping, as specified in EN 301 192 [38], and duly signalled as such, in compliance with EN 300 468 [16]. A service may contain one or more Data Pipes carrying files via FDP.

H.2.2 Encapsulation of FDP in Data Pipes

Files are broadcast with the help of the FDP messages, as specified in clauses H.3 and H.4 below. Each of these FDP messages is carried in a DVB Data Pipe, i.e. the FDP Messages are directly encapsulated into the payload of the MPEG-2 Transport Stream packets.

The start of an FDP Message may or may not be aligned with the start of the Transport Stream packet payload.

The `payload_unit_start_indicator` shall be used for FDP Messages in the same way as specified in ISO/IEC 13818-1 [46] for PSI sections, i.e. with the following significance: if the Transport Stream packet carries the first byte of an FDP Message, the `payload_unit_start_indicator` value shall be '1', indicating that the first byte of the payload of this Transport Stream packet carries the pointer_field. If the Transport Stream packet does not carry the first byte of an FDP Message, the `payload_unit_start_indicator` value shall be '0', indicating that there is no `pointer_field` in the payload.

The `pointer_field` is used as specified in ISO/IEC 13818-1 [46] to indicate the position of the first byte of the first FDP Message starting in the Transport Stream Packet.

Only one file shall be carried with FDP in one given Data Pipe at a given moment. However, broadcasting files simultaneously with FDP is possible, by using different Data Pipes.

H.2.3 File identification

In FDP, files are identified uniquely using the `organisation_id` and a `file_id`, which are both 32 bit identifiers present in the header of all FDP Messages. This means that an `organisation_id` and `file_id` combination cannot correspond to more than one file.

Consequently, the terminal shall treat multiple files identified with the same `organisation_id` and `file_id` values as being the same file being broadcast multiple times.

H.2.4 Referring to files using URLs

The location where a file is carried via FDP can be referred to using a URL. This URL has the same form as a `dvb:` URL (as specified in TS 102 851 [10] and IETF RFC 3986 [27]) with the only difference being that the URI Scheme is `fdp:` instead of `dvb:`. Such URLs are referred to as "FDP URLs".

FDP URLs are defined to have the form:

`fdp://<original_network_id>.<transport_stream_id>.<service_id>.<component_tag>/<organisation_id>/<file_id>`

where `file_id` and `organisation_id` are the 32 bit identifiers defined in clause H.2.3 encoded as `hex_string` as defined in TS 102 851 [10] but with no leading zeros. All other parts are as defined in TS 102 851 [3].

H.3 File segmentation and broadcasting

H.3.1 File segmentation

Each file shall be divided into segments (the File Segments). These segments shall be of equal size except for the last File Segment which may be smaller. The size of these segments shall be specified in the `file_segment_size` field of the Initialization Message of the file, which precedes the Data Messages carrying these segments.

Optionally, in addition to the File Segments (containing the actual data of the file), the Data Messages may also carry segments containing error correction data that the terminal can use to reconstruct the file in case some File Segments have not been properly received (the FEC Segments). When present, these segments shall be of equal size except for the last FEC Segment which may be smaller. The size of these segments shall be specified in the `FEC_segment_size` field of the Initialization Message of the file, which precedes the Data Messages carrying these segments.

The `message_type` field indicates to the terminal whether a Data Message carries a File Segment or a FEC Segment.

All File Segments shall be numbered contiguously and in order, starting at the value 0x00000000.

The number of File Segments for a given file can be calculated as follows:

$$N_{file} = INT ((file\_size - 1) / file\_segment\_size) + 1$$

Consequently, all File Segments have a `segment_ number` in the range $[0 : N_{file} - 1]$.

When present, FEC segments shall be numbered contiguously and in order, starting at the value 0x00000000.

The number of FEC Segments for a given file can be calculated as follows:

$$N_{fec} = INT ((FEC\_size - 1) / FEC\_segment\_size) + 1$$

Consequently, when present, FEC Segments have a `segment_ number` in the range $[0 : N_{fec} - 1]$.

The file segmentation and the message sequences are illustrated in the figure H.1.

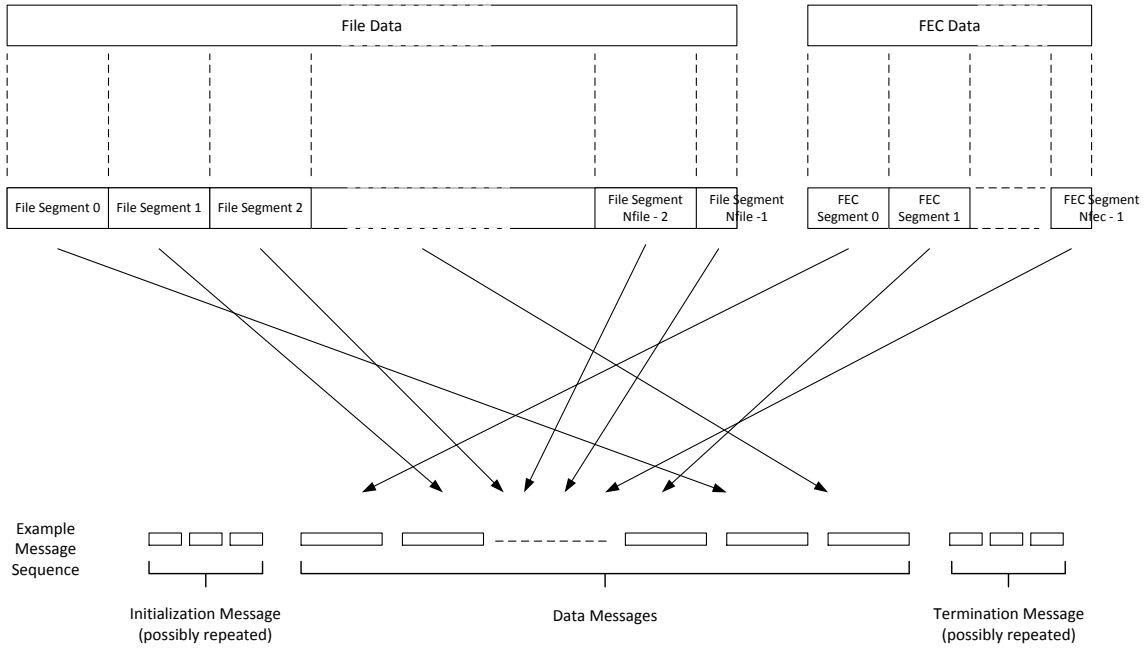


Figure H.1: FDP message sequence

The use of FEC is optional, and is not specified in the present document. FEC Segments can only be sent in addition to File Segments. All File Segments shall be broadcast, regardless of whether FEC segments are broadcast or not, to allow terminals ignoring FEC segments to be able to reconstruct files from the received File Segments.

H.3.2 Message sequence

The timing of the broadcast of files using FDP is described by the `<availabilityWindow>` element of the Content Access Download Descriptor (as defined in the OIPF DAE specification [1]).

For a file download to take place, the following message sequence has to occur within the availabilityWindow period:

- Initialization Message (possibly repeated several times).
- Data Messages, carrying the File Segments and possibly also FEC Segments.
- Termination Message (possibly repeated several times).

NOTE: One or more repetitions of the Initialization Message may be broadcast after the first Data Message.

Within the `availabilityWindow` period, the following rules shall apply:

- The above message sequence shall not occur more than once per availabilityWindow period.
- There shall be at least one Initialization Message sent between the start of the availabilityWindow period and the first Data Message.
- There shall be at least one Termination Message sent between the last Data Message and the end of the availabilityWindow period.
- There shall not be any Initialization Messages sent after the first Termination Message
- All Data Messages making up the file shall be sent between the first Initialization Message and the first Termination Message. However, during this period Data Messages may be sent in any order and each Data Message may be sent more than once.

Each message contains a `CRC_32`. The terminal shall check the integrity of each received message by checking the CRC value as specified in annex A of ISO/IEC 13818-1 [46].

H.3.3 Repeated broadcasts of file segments

When segments of the same file (according to the criterion specified in clause H.2.3) are broadcast multiple times, each of these segments shall be identical across all repetitions. Therefore, it shall be possible for the terminal to reconstruct the file by reassembling segments obtained from different broadcasts of same file.

The multiple broadcasts may be carried in the same Data Pipe (same component of the same data broadcast service), or in a different Data Pipe (e.g. in another service, located on a different Transport Stream).

In case of multiple broadcasts of a file, the following shall apply:

- When the terminal has received a subset of the File Segments related to a file, the terminal shall not discard those File Segments but shall attempt to receive the missing File Segments during a subsequent broadcast.
- When the terminal has received a set of File Segments and FEC Segments which permits the successful reconstruction of the file, the terminal shall deem the download as completed, disregard all subsequent messages relating to this file, and free-up the corresponding resources.

H.3.4 File segment recovery

The Initialization Message may provide a Recovery URL, which is associated with the file it relates to. This Recovery URL indicates the location where the terminal may retrieve the File Segments which have not been received successfully. The use of the recovery URL by the terminal is optional.

To retrieve a specific File Segment using this Recovery URL, the terminal shall issue an HTTP GET with the URL formed by the concatenation of the Recovery URL with <segment_number> (as an 8-character hex string).

EXAMPLE:

If the Recovery URL supplied for a specific file is:

`http://recovery_server.service_provider.com/recovery/FileXYZ/,`

the terminal can retrieve the File Segment number 0x01234567 of this file at the following URL:

`http://recovery_server.service_provider.com/recovery/FileXYZ/01234567.`

The terminal shall only use this mechanism in cases where a subset of the File Segments of a file has been successfully received, but this subset is not sufficient for the file to be completely reconstructed and there is no subsequent availabilityWindow known to the terminal during which missing segments could be received. In such cases:

- If a Recovery URL is associated with the file, the terminal may attempt to fetch missing segments via broadband using this URL as described above. If so, the first request to this URL shall occur only after the receipt of a Termination Message within the last known `availabilityWindow` associated to the file or after the end of this last known `availabilityWindow`, whichever earlier. From this event, the terminal shall wait a random duration between zero and the value of the `time_dispersion` field specified in the Initialization Message (see clause H.4.2) before performing this first request. This randomization is required, in order to avoid large numbers of terminals contacting the recovery server simultaneously, resulting possibly in server overloads.
- If for any reason, such a recovery is not attempted, or is unsuccessful, the terminal shall consider the download of this file as completed with errors or failed (depending on the value of `discard_file_on_error_flag` as described in clause H.4.2) and report accordingly to the application.

H.4 Syntax and semantics of FDP messages

H.4.1 Message types

The FDP messages can be of different types but all include the same header information. The Message Type is indicated in the header of each message as an 8-bit field, with a value as defined in Table H.1.

Table H.1: FDP message types

Value	Type of message
0x00	Reserved
0x01	Initialization Message
0x02-0x10	Reserved
0x11	Data Message containing a File Segment (see clause H.3.1)
0x12	Data Message containing a FEC Segment (see clause H.3.1)
0x13-0x20	Reserved
0x21	Termination Message
0x22-0xFF	Reserved

H.4.2 Initialization Message

The Initialization Message shall have the following syntax.

Table H.2: Initialization message

Syntax	Number of bits	Identifier
initialization_message() {		
protocol_version	8	uimsbf
message_type	8	uimsbf
message_length	16	uimsbf
organisation_id	32	uimsbf
file_id	32	uimsbf
file_size	48	uimsbf
file_segment_size	16	uimsbf
FEC_size	48	uimsbf
FEC_segment_size	16	uimsbf
max_recovery_requests	20	uimsbf
min_time_between_requests	12	uimsbf
time_dispersion	16	uimbsf
discard_file_on_error_flag	1	bslbf
reserved_future_use	7	bslbf
recovery_URL_length	8	uimsbf
for (i=0 ; i < N ; i++) {		
recovery_URL_char	8	uimsbf
}		
private_data_length	8	uimsbf
for (i=0 ; i < M ; i++) {		
private_data_byte	8	uimsbf
}		
CRC_32	32	rpchof
}		

Semantics for the Initialization Message:

protocol_version: This field shall have the value 0x01. Terminals shall ignore all messages that have a different value in this field.

message_type: In Initialization Messages, this field shall have the value 0x01.

message_length: This field specifies the number of bytes of the message, starting immediately following the `message_length` field and including the CRC.

organisation_id: This field is the `organisation_id` as defined in TS 102 851 [3]. Along with the `file_id`, it forms a tuple which uniquely identifies the file being broadcast (see clause H.2.3).

file_id: This field is the identifier of the file as defined in clause H.2.3. Along with the `organisation_id`, it forms a tuple which uniquely identifies the file being broadcast.

file_size: This field gives the total size (in bytes) of the file.

file_segment_size: This field gives the number of data bytes in the File Segments, except for the last File Segment of the file which may be smaller (see clause H.3.1).

FEC_size: This field gives the total size (in bytes) of the FEC data block (see clause H.3.1). If there are no FEC segments transmitted, this field shall have the value zero.

FEC_segment_size: This field gives the number of data bytes in the FEC Segments sent in addition to the File Segments, except for the last FEC Segment which may be smaller (see clause H.3.1). If `FEC_size` equals zero, this field has no meaning.

discard_file_on_error_flag: This field is a 1-bit flag indicating how the terminal should behave when a file cannot be reconstructed without any error. A value of '0' indicates that the terminal shall keep the file even when there are uncorrected errors (erroneous or missing File Segments). In such a case, the Download shall be reported as successfully completed to the application, and the presence of errors shall be reported by using the `errorLevel` property of the `Download` class (see clause A.2.11). A value of '1' indicates that the terminal shall not keep the file when there are uncorrected errors (erroneous or missing File Segments). In such a case, the incomplete or erroneous file shall be deleted, and the Download shall be reported to the application as "failed".

max_recovery_requests: This is a 20-bit field indicating the maximum number of recovery requests each terminal may make to the Recovery URL (see clause H.3.4) to recover File Segments of this file. This includes any retries a terminal may make in response to failing requests. A value of '0' indicates that there is no limit specified by this message. If there is no Recovery URL provided in this message, this field may be ignored.

min_time_between _requests: This is a 12-bit field indicating the minimum time (in seconds) that shall separate two consecutive requests made by a terminal to the `Recovery_URL`. In case there is no Recovery URL provided in this message, this field may be ignored.

time_dispersion: This is a 16-bit field indicating the range (in seconds) of the time dispersion that shall be applied by terminals to randomize the time at which the terminal will perform the first attempt to retrieve missing segments using the Recovery URL (as described in clause H.3.4). A value of '0' indicates that no time dispersion is required. If there is no Recovery URL provided in this message, this field may be ignored.

recovery_URL_length: This field gives the total length in bytes of the text string forming the Recovery URL (see clause H.3.4). If no Recovery URL is provided in this message, this field shall have the value zero.

recovery_URL char: Character of the text string forming the Recovery URL (see clause H.3.4).

private_data_length: This field gives the total length in bytes of the following loop containing private data.

private_data_byte: This is an 8-bit field, the value of which is privately defined.

CRC_32: This is a 32 bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in annex B of ISO/IEC 13818-1 [46] after processing the entire message.

H.4.3 Data Message

A Data Message can be used to carry either a File Segment or a FEC Segment (see clause H.3.1 for details).

The Data Message shall have the following syntax.

Table H.3: Data message

Syntax	Number of bits	Identifier
data_message() {		
protocol_version	8	uimsbf
message_type	8	uimsbf
message_length	16	uimsbf
organisation_id	32	uimsbf
file_id	32	uimsbf
segment_number	32	uimsbf
data_length	16	uimsbf
for (i=0 ; i < N ; i++) {		
data_byte	8	uimsbf
}		
CRC_32	32	rpchof
}		

Semantics for the Data Message:

protocol_version: This field shall have the value 0x01. Terminals shall ignore all messages that have a different value in this field.

message_type: In Data Messages carrying File Segments, this field shall have the value 0x11. In Data Messages carrying FEC Segments, this field shall have the value 0x12.

message_length: This field specifies the number of bytes of the message, starting immediately following the `message_length` field and including the CRC.

organisation_id: This field is the `organisation_id` as defined in TS 102 809 [3]. Along with the `file_id`, it forms a tuple which uniquely identifies the file being broadcast (see clause H.2.3).

file_id: This field is the Identifier of the file as defined in clause H.2.3. Along with the `organisation_id`, it forms a tuple which uniquely identifies the file being broadcast.

segment_number: This 32 bit field gives the number of the segment, allowing the Terminal to re-order the segments to reconstruct the file. The first segment of the file shall have the number 0x00000000. See clause H.3.1 for details.

data_length: This field gives the total length in bytes of the following loop containing the message data

data_byte: This is an 8-bit field, constituting the content of the File Segment or the FEC segment. The number of data_byte fields in a Data Message shall be equal to the `file_segment_size` or `FEC_segment_size` field, as appropriate, carried in the Initialization Message within this availabilityWindow period.

CRC_32: This is a 32 bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in annex B of ISO/IEC 13818-1 [46] after processing the entire message.

H.4.4 Termination Message

The Termination Message shall have the following syntax.

Table H.4: Termination message

Syntax	Number of bits	Identifier
termination_message() {		
protocol_version	8	uimsbf
message_type	8	uimsbf
message_length	16	uimsbf
organisation_id	32	uimsbf
file_id	32	uimsbf
private_data_length	8	uimsbf
for (i=0 ; i < M ; i++) {		bslbf
private_data_byte	8	uimsbf
}		
CRC_32	32	rpchof
}		

Semantics for the Termination Message:

protocol_version: This field shall have the value 0x01. Terminals shall ignore all messages that have a different value in this field.

message_type: In Termination Messages, this field shall have the value 0x21.

message_length: This field specifies the number of bytes of the message, starting immediately following the `message_length` field and including the CRC.

organisation_id: This field is the `organisation_id` as defined in TS 102 809. Along with the `file_id`, it forms a tuple which uniquely identifies the file being broadcast (see clause H.2.3).

file_id: This field is the Identifier of the file as defined in clause H.2.3. Along with the `organisation_id`, it forms a tuple which uniquely identifies the file being broadcast.

private_data_length: This field gives the total length in bytes of the following loop containing private data

private_data_byte: This is an 8-bit field, the value of which is privately defined.

CRC_32: This is a 32 bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in annex B of ISO/IEC 13818-1 [46] after processing the entire message.

Annex I (informative): Push-VoD services

I.1 Introduction

Terminals which support the "download feature" provide an enhanced content download API which allows for implementing push-VoD services. Following this concept an HbbTV® application schedules the download of movies or other audio-visual content prior to its presentation. Content is downloaded from the broadcast channel. The application should schedule content downloads automatically and without any user interaction. Once the content is completely downloaded it can be offered for playback to provide an instantaneous and error-free VoD experience. Push-VoD provides highest quality of service even for content of high data volume (HD, 3D, Ultra HD) and even if a high bandwidth broadband channel is not available.

I.2 Level of trust

APIs used for push-VoD services are trusted. Therefore, the service provider should ensure that its application is trusted on the target terminals.

I.3 Protocols

I.3.1 Broadcast protocol

The protocol for push-VOD services is FDP as defined in annex H.

I.3.2 Download protocol

A download is described by a Content Access Download Descriptor (CADD) as defined in annex E of OIPF DAE with the clarifications as described in clause 7.3.1.5.1 of the present document.

I.3.3 Sources

The source of the file data is defined by an FDP URL as defined in clause H.2.4.

I.4 Application features

I.4.1 Overview on application features

In general, a Push-VoD application will provide at least the following user interfaces:

- Advertising the service.
- Subscription / Unsubscription to the service.
- Content browsing and management.
- Content Playback.

Additionally, a push-VoD application will necessarily implement at least the following functions in the background:

- Schedule downloads.
- Delete or cancel old downloads.

Applications may also reserve hard disk space for push-VoD content.

Applications will implement these functions using the Content Download API as defined in clause 7.4 of the OIPF DAE specification [1].

I.4.2 Hard disk space reservation

To ensure that there will be space on the hard disk for scheduled downloads, the application should use the Content Download API to reserve a part of the hard disk for its push-VoD content. Applications from the same organisation (as indicated by the application's organisation ID) can only reserve one part of the hard disk. Hard disk space reservation is usually done when the user subscribes to the service, and accepts its conditions.

To reach a wide range of coverage the content provider should carefully decide about the size of the reserved space. It might be limited due to hardware capabilities or due to the hard disk being used for PVR recordings or other applications. The application should not reserve more space than needed.

The Terminal decides whether or not to reserve storage space in response to a request from an application. HbbTV® intentionally does not define the criteria that are used for making this decision. Some examples of possible criteria that may be used include the following:

- The order in which requests to reserve space were made.
- Space being available.
- Asking the end-user.
- A commercial agreement between the terminal manufacturer and the service provider making the request.
- The presence of the calling application on a white list of applications maintained by the terminal manufacturer.
- The absence of the calling application from a black list of applications maintained by the terminal manufacturer.

If the Terminal refuses the requested reservation, the application should evaluate the return value of the API function and should inform the user accordingly.

I.4.3 Hard disk deallocation

A push-VoD application should provide the possibility to free some or all of the storage space that it may have reserved, allowing the user to recover it for other purposes. Thus, if the user selects such an option, the application should:

- Delete all completed and in-progress downloads.
- Cancel all scheduled downloads.
- Free its storage space.

Unsubscription from a push-VoD service should necessarily result in the freeing of the reserved space.

Additionally, the Terminal UI may also provide means to free storage space, independently from any application.

I.5 Content management

I.5.1 Content schedule

The content provider should plan its content schedule according to the reserved space.

The application is responsible for deleting expired content (or for flagging which content items can be deleted automatically by the terminal) and to cancel outdated downloads. The criteria for deciding which downloads should be deleted are defined by the application. This should be done prior to scheduling new downloads if necessary.

I.5.2 Play-out

The content provider should be aware that there may be competing access resources of the Terminal such as tuners (e.g. for other push-VoD services, for TV viewing, or for PVR recordings). Therefore, the content provider should provide several availability windows for each content item.

I.6 Playback

The application should provide a user interface for browsing and playing the downloaded content. Additionally, the Terminal UI may provide access to downloaded content. In that case the Terminal should only offer content that is already completely downloaded and ready for playback. Downloads scheduled for the future or ongoing downloads should not be shown.

HbbTV® applications from one organization are not able to access the content downloaded by an application from a different organization.

Annex J (informative): Advert insertion guidance for content providers

The present document provides support for dynamic insertion of advertising using multiple HTML5 media elements. This clause provides guidance to content providers on how to use these capabilities.

Application authors should write their applications assuming that at any given time, only one `<video>` or `<audio>` element can be in the PLAYING state. If multiple media elements exist within the DOM, the transition to the PLAYING state of one media element might cause all other media elements to transition to the PAUSED state.

As per clause 9.6.1, how the terminal renders `<video>` elements that are not in the PLAYING state is undefined. This means that if multiple `<video>` elements exist within the DOM it is possible that the `<video>` element that is in the PLAYING state might be obscured by one or more of the other `<video>` elements that are not in the PLAYING state. To maximise interoperability between HTML5 environments, it is recommended that application authors ensure that any `<video>` element that is not required to be actively presenting content is explicitly hidden. Using "`display:none`" for media elements that are in the PAUSED state is a method to achieve this recommendation.

When creating a `<video>` element for prefetching, the computed CSS of this element should have the `display` property set to `none`.

When creating a media element for prefetching, the recommended order of actions is:

1) Create the media element, for example by using `document.createElement("video")`.
2) Set the CSS `display` property to `none`, either by directly setting this property on the element or via a CSS class with a suitable CSS rule.
3) Add the media element to the DOM.
4) Call the `load()` function of the media element.

When switching between `<video>` elements, the recommended order of actions is:

1) Set the `display` CSS property of the pre-fetched video to `block`.
2) Pause the currently playing media element, using the `pause()` function.
3) Start playback of the pre-fetched media, using the `play()` function.
4) Set the `display` CSS property of the previous media element to `none`.
5) If the previous media element is no longer required, remove it from the DOM.

If a `<video>` element is to be paused and resumed later on, it needs to be kept in the DOM. When resuming a `<video>` element, the terminal might have discarded previously decoded key frames, which might delay the start of presentation until the next random access point in the stream is reached.

Detecting when to perform the switch between `<video>` elements can be implemented in a variety of ways, including:

- Listening to "`timeupdate`" events from the currently playing media element.
- Polling the "`currentTime`" attribute of the currently playing media element.
- Listening to "`cuechange`", "`enter`" and "`exit`" events of a `TextTrack` on the currently playing media element.

For accurate timing, it is possible to combine these techniques. For example, a "`timeupdate`" event can be used to discover the approximate playback position and then, 500 ms before the ad break, the application can switch to polling the "`currentTime`" attribute of the currently playing media element using `setTimeout()` or `setInterval()`. Note that the HTML5 Recommendation [54] requires a terminal to emit a "`timeupdate`" event for this media element at least every 250 ms, therefore relying on the "`timeupdate`" event might not provide sufficient accuracy for ad insertion by itself.

Each media element might consume significant amounts of memory on the terminal, because several fragments of each stream might be downloaded in to system memory before the "canplay" event is fired. Application authors should limit the number of media elements that exist in the DOM to three, and should be careful to make sure they remove all references to a media element when removing it from the DOM.

The current best practice is to remove all listeners from the media element, set the src attribute to an empty string, delete this attribute, call the load() function on the <video> element and then remove the media element from the DOM.

```
var last;
videoElement.pause();
while (last = videoElement.lastChild) {
    videoElement.removeChild(last);
}
videoElement.setAttribute("src","");
try{
    videoElement.removeAttribute("src");
} catch(e){
}
videoElement.load();
videoElement.parentNode.removeChild(videoElement);
videoElement=null;
```

Care needs to be taken to avoid accidentally keeping a reference to a media element within the closure scope of a function. Such a reference would cause the JavaScript virtual machine to have a reference to the media element that would stop the garbage collector releasing the media element's resources.

There is no prioritisation between media elements, which means that the prefetching of one media element might impact the media element that is currently playing. If the currently playing video uses adaptive bitrate control, the prefetching of one media element might cause the currently playing video to drop to a lower bitrate representation. Typically two to three fragments are downloaded of a pre-fetched video.

When implementing the soft-partition use case, the buffered property of the "long-form" video element can be used to discover when sufficient data has been downloaded to reach the next advertising break. Once sufficient content has been downloaded to reach the next advertising break, the preload property can set to "none" or "metadata", to provide a hint that retrieval of data should be stopped. If possible, delay pre-fetching adverts until the buffered property of the "long-form" video element indicates that sufficient data has been downloaded to reach the next advertising break. When the last advert in the advertising break is playing, the preload property of the "long-form" content should be set to "auto".

If the preload attribute of a media element is set to "metadata", the terminal should reduce the amount of data that the <video> element downloads when the load() function is called. If a <video> element was pre-fetched with the preload attribute set to "metadata", it is recommended to change this to "auto" once video playback has started. The reason for this recommendation is that a media element with preload="metadata" in the PLAYING state is defined in the HTML5 Recommendation [54] to indicate to the terminal that it should minimise its bandwidth consumption when playing this media. For streams using adaptive bitrate control, this might cause an unnecessarily conservative bitrate adaptation to be chosen.

The following table summarises how the HTML5 Recommendation [54] defines how an application can influence the buffering decisions of the terminal:

PLAYING/PAUSED state	Preload property	Acquisition state
PAUSED load() has not been called	n/a	Not playing, not acquiring
PAUSED load() has been called	"metadata"	Not playing, acquiring at "reduced rate"
PAUSED load() has been called	"auto"	Not playing, acquiring at "any rate"
PLAYING	"metadata"	Playing, acquiring at "reduced rate"
PLAYING	"auto"	Playing, acquiring at "any rate"
PLAYING	"none"	Playing, not acquiring

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

K.1 Introduction (informative)

This annex defines how an HbbTV terminal is able to delegate requests to play content that originate from an HbbTV application to a media player in a CICAM. Specifically it defines the integration between the video element API from HTML5 [54] and the "Host-initiated playback mode" of the CICAM player resource from TS 103 205 [37].

NOTE: This annex only applies to the HTML5 video element and not to the HTML5 audio element.

Behaviour of either the HTML5 video element or the HbbTV terminal implementation of the CI+ protocol that is not related to the integration between them is outside the scope of this annex.

When the CICAM player resource is used in "Host-initiated playback mode":

- a media player in the CICAM is used to present content controlled by the HTML5 video element instead of the media player in the HbbTV terminal that is normally used
- the HbbTV application controls media playback through the methods and properties of the HTML5 video element API which are then translated by the HbbTV terminal to the messages (called "APDU"s) supported by the CICAM player resource
- the media player in the CICAM handles IP delivery protocols in place of the HbbTV terminal, and returns the content to HbbTV terminal as a SPTS.

Tables K.1 and K.2 below summarise this integration firstly from the viewpoint of the CICAM (by listing the APDUs defined for the CICAM player resource) and secondly from the point of view of the HTML page (by listing the HTML5 video element methods and attributes).

Table K.1: CICAM Player resource APDUs

APDU	Direction	APDU usage	Reference
CICAM_player_verify_req	Host -> CICAM	Requests CICAM to check to see if the content can be consumed.	K.3
CICAM_player_verify_reply	CICAM -> Host	Signals if it is able to consume the content.	K.3
CICAM_player_capabilities_req	CICAM -> Host	Enables the CICAM to acquire codec capabilities from the Host.	Outside the scope of this annex.
CICAM_player_capabilities_reply	Host -> CICAM	The set of codecs that the Host supports.	Outside the scope of this annex.
CICAM_player_start_req	CICAM -> Host	Request to start player session	K.3
CICAM_player_start_reply	Host -> CICAM	Response to start request	K.3
CICAM_player_play_req	Host -> CICAM	Request by host to play a content item.	K.3
CICAM_player_status_error	CICAM -> Host	Signal critical error to Host	K.7
CICAM_player_control_req	Host -> CICAM	Instructs CICAM to seek to a given position and/or change playback speed.	K.8, K.9
CICAM_player_info_req	Host -> CICAM	Request for play speed, position, and duration	K.4, K.8, K.9
CICAM_player_info_reply	CICAM -> Host	Play speed, position and duration	K.4, K.8, K.9
CICAM_player_stop	Host -> CICAM	Instruct CICAM to terminate playback	K.4, K.5
CICAM_player_end	CICAM -> Host	Free playback session on Host	K.5
CICAM_player_asset_end	CICAM -> Host	Informs Host that end of asset has been reached	K.6
CICAM_player_update_req	CICAM -> Host	CICAM requests to provide updated components	K.10
CICAM_player_update_reply	Host -> CICAM	Host responds to player update request	Outside the scope of this annex.

Table K.2: HTML5 video element attributes and methods

Name	Description	Reference
readonly attribute MediaError? Error	Last error in media playback – one of aborted, network error, decode error, content not supported.	K.7
attribute DOMString src	URL of content item, can be written to set a new content item	K.2, K.5
readonly attribute DOMString currentSrc	URL of current content item or empty string	K.2
attribute DOMString crossOrigin	Use of cookies (etc) for cross-origin requests	K.11
readonly attribute unsigned short networkState	One of not initialised, initialised but not using network, loading data, initialising.	K.4
attribute DOMString preload	Hint of extent to which content item may be preloaded (none, metadata only, optimistic download appropriate).	Outside the scope of this annex.
readonly attribute TimeRanges buffered;	Returns the part of the content item that is buffered locally.	K.4
void load()	Reload the media element after changing the source or other properties.	K.3, K.5
CanPlayTypeEnum canPlayType(DOMString type)	Test if the MIME type is one that can be decoded, cannot be decoded or cannot be determined.	K.11
readonly attribute unsigned short readyState	Return how much of the metadata & data for the content item has been loaded – nothing, just metadata, some data but not enough to start playing, enough data to start playing.	K.4
readonly attribute boolean seeking	Indicates that an asynchronous seek is in progress.	K.9
attribute double currentTime	The current play position. Writing to this starts a seek.	K.4, K.9
readonly attribute unrestricted double duration	The time of the end of the content item.	K.4, K.9
Date getStartDate()	An explicit date and time corresponding to the zero time in the media timeline.	K.11
readonly attribute boolean paused	Indicates whether playback is paused.	K.8
attribute double defaultPlaybackRate	Default playback speed (“The defaultPlaybackRate is used by the user agent when it exposes a user interface to the user.”)	Outside the scope of this annex.
attribute double playbackRate	On reading, returns the current playback rate. On writing, attempts to change the playback rate.	K.8
readonly attribute TimeRanges played	represents the ranges of points on the media timeline of the media resource reached through the usual monotonic increase of the current playback position during normal playback	K.4
readonly attribute TimeRanges seekable	represents the ranges of the media resource, if any, that the user agent is able to seek to, at the time the attribute is evaluated.	K.4, K.9
readonly attribute boolean ended	Indicates that playback reached the end of content.	K.6
attribute boolean autoplay	Automatically begin playback of the media resource as soon as enough data is available to do so without stopping/pausing.	Outside the scope of this annex.
attribute boolean loop	If set, automatically seek back to the start of the media resource upon reaching the end and play from there.	Outside the scope of this annex.
void play()	Play the content	K.3, K.5
void pause()	Pause the content presentation	K.8
attribute boolean controls	Indicates whether the HTML page has provided a UI for control of media playback otherwise the user agent should do this.	Outside the scope of this annex.
attribute double volume	Audio playback volume between 0.0 and 1.0.	Outside the scope of this annex.
attribute boolean muted	On reading, indicate if audio is muted. On writing mute or unmute audio.	Outside the scope of this annex.
attribute boolean defaultMuted	Indicate that audio should default to muted when presentation starts.	Outside the scope of this annex.
readonly attribute AudioTrackList audioTracks;	List of audio tracks in the content item	K.10
readonly attribute VideoTrackList videoTracks	List of video tracks in the content item	K.10
readonly attribute TextTrackList textTracks	List of subtitle and other tracks in the content item	K.10
TextTrack	Creates and returns a new TextTrack object, which is also	K.11

addTextTrack(TextTrackKind kind, optional DOMString label, optional DOMString language)	added to the media element's list of text tracks.	

NOTE: If the communication interface between HbbTV application and CICAM defined in this annex is not rich enough then, subject to support by the CICAM, the message passing mechanism between the 'application/oipfDrmAgent' embedded object and the CI Plus protocol defined in clause 11.4.1 of the present document can be used in addition.

K.2 Determining when to use CICAM player mode

When a Content Access Streaming Descriptor including the `CICAMPlayerPreferred` element is used with an HTML5 `video` element then the HbbTV terminal shall attempt to present the content referenced by the `contentURL` element using CICAM player mode. The HbbTV terminal shall not attempt to use CICAM player mode under other circumstances – e.g A/V control object, HTML5 video element with `src` or `source` directly referencing content, HTML5 video element with `src` or `source` referencing Content Access Streaming Descriptor without `CICAMPlayerPreferred` element or HTML5 video element with `src` or `source` referencing DASH MPD.

K.3 Starting CICAM player

The process for session initialisation of Host-Initiated playback as described in clause 8.3.2 of TS 103 205 [37] shall apply for an HTML5 video element where it has been determined that a CICAM player will be used (according to clause K.2) and either the `load()` or `play()` methods are called or the `autoplay` property is set and takes effect. The URL from the `<ContentURL>` element shall be wrapped in a ServiceLocation and passed to the CICAM in the `CICAM_player_play_req()` and/or `CICAM_player_verify_req()` APDUs. When available in the Content Access Streaming Descriptor, the `DRMControlInformation` element obtained from the Content Access Streaming Descriptor shall also be wrapped in the ServiceLocation passed to the CICAM. The ServiceLocation shall have no ContentAttributes and the priority attribute shall be set to zero.

K.4 During CICAM player use

With reference to the sequence diagram for host-initiated playback in clause 8.7 of TS 103 205 [37], the `networkState` attribute shall be updated as defined by the resource selection and resource fetch algorithms in the HTML5 specification except as follows;

- When the `input_max_bitrate` requested by the CICAM is not zero:
  - The value `NETWORK_IDLE` shall be returned between when the HbbTV terminal has received the `CICAM_player_start_req()` APDU from the CICAM and when the HbbTV terminal sends the `comms_info_reply()` APDU to the CICAM.
  - The value `NETWORK_LOADING` shall be returned between when the HbbTV terminal sends the `comms_info_reply()` APDU to the CICAM until the HbbTV terminal sends the `CICAM_player_stop()` APDU to the CICAM.
- When the `input_max_bitrate` requested by the CICAM is zero (CICAM uses its own connectivity):
  - The value `NETWORK_LOADING` shall be returned between when the HbbTV terminal has received the `CICAM_player_start_req()` APDU from the CICAM until the HbbTV terminal sends the `CICAM_player_stop()` APDU to the CICAM.

The value of the `readyState` attribute shall as follows;

- `HAVE_NOTHING` when the `networkState` attribute has values other than `NETWORK_LOADING`.
- `HAVE_ENOUGH_DATA` when the `networkState` attribute has the value `NETWORK_LOADING`.

The `buffered` attribute shall return a `TimeRanges` object as follows;

- If the duration of the content is known (i.e. the `duration` field in the last `CICAM_player_info_reply()` APDU was not 0xFFFFFFFF) then the `TimeRanges` object shall represent the full duration of the content item as defined for the value of the `duration` attribute.

- If the duration of the content is not known (i.e. the `duration` field in the last `CICAM_player_info_reply()` ADPU was 0xFFFFFFFF) then the `TimeRanges` object shall represent 5 seconds either side of the current play position.

The `played` attribute shall be calculated by the HbbTV terminal as defined in the HTML5 specification using the values of the current playback position.

An HbbTV application reading the current playback position (e.g. `currentTime` property) shall be handled as follows;

- HbbTV terminals shall maintain a local copy of the current playback position as defined in the HTML5 specification [54] and update this as playback proceeds and return this to applications.

- HbbTV terminals shall poll the CICAM player current play position by sending the `CICAM_player_info_req()` APDU. This shall be done at intervals of not more than 30 seconds.

- If the resulting `CICAM_player_info_reply()` APDU from the CICAM includes a position that is not 0xFFFFFFFF then the terminal shall re-synchronise the local copy of the current playback position from this value and then update it the value as playback proceeds until the next poll.

- HbbTV terminals shall not send a `CICAM_player_info_req()` APDU each time an application reads the current playback position and then block until a `CICAM_player_info_reply()` APDU is received.

The HbbTV terminal shall maintain a local copy of the duration of the content and return this when the duration attribute is read. Each time a `CICAM_player_info_reply()` APDU is received from the CICAM, if the value of the duration field in that APDU is not 0xFFFFFFFF then the local copy of the duration shall be updated if different. Each time the local copy is updated then a `TimeRanges` object shall be generated to be returned from the `seekable` property which shall contain a single range whose start is zero and whose end is the duration returned from the CICAM.

If the content item referenced from the video element is changed to a different one and the conditions in clause K.2 above apply for the updated content item, then the CICAM player session shall be stopped and a new CICAM player session shall be started as defined in clause K.3.

K.5 Stopping CICAM player use

An HTML5 video element shall cease using a CICAM player, according to clause 8.3.2 and 8.6.3 of TS 103 205[37], if any of the following apply;

- The content item referenced from the video element is changed to a different one such that none of the conditions in clause K.2 above apply to the new content item and either the `load()` or `play()` methods are called.

- The `src` attribute of the `video` element is set to empty and the `load()` method is called

- The HTML5 video element has released the video and audio decoder resources as defined in clause 9.6.2 of the present document

- The HTML5 video element is garbage collected following removal from the DOM tree

K.6 Play to end of content

If the CICAM player sends a `CICAM_player_asset_end()` APDU to the HbbTV terminal with the `beginning` field set to '0' then the procedure for "When the current playback position reaches the end of the media resource when the direction of playback is forwards" shall be followed as defined in clause 4.7.10.8 of the HTML5 specification [54].

If the CICAM player sends a `CICAM_player_asset_end()` APDU to the HbbTV terminal with the `beginning` field set to '1' then the procedure for "When the current playback position reaches the earliest possible position of the media resource when the direction of playback is backwards" shall be followed as defined in clause 4.7.10.8 of the HTML5 specification [54].

K.7 Errors

When the HbbTV terminal receives a `CICAM_player_status_error()` APDU from the CICAM, it shall set the `error` attribute with the mapping shown in Table K.3:

Table K.3 - Mapping of error values

player_status from the CICAM		MediaError? Error	
0x01	Error - content play is not possible (e.g. unsupported content format or protocol)	4	`MEDIA_ERR_SRC_NOT_SUPPORTED`
0x02	Error - unrecoverable error	4	`MEDIA_ERR_SRC_NOT_SUPPORTED`
0x03	Error - content blocked (e.g. no content license available)	3	`MEDIA_ERR_DECODE`

Since the HbbTV terminal has no information about the CICAM buffering or IP connectivity, IP connectivity issues and errors in the HbbTV terminal shall not be directly reported as errors via the HTML5 video element.

K.8 Play speed

An application setting the playbackRate property shall result in the terminal sending a `CICAM_player_control_req()` APDU to the CICAM with the `command` element set to 0x02 and the `Speed` element being the value set by the application converted from decimal to "hundredths of the nominal speed".

An application calling the `pause()` method shall result in the terminal sending a `CICAM_player_control_req()` APDU to the CICAM with the `command` element set to 0x02 and the `Speed` element being forced to 0.

The terminal shall periodically query the CICAM player current speed by sending the `CICAM_player_info_req()` APDU. The terminal shall maintain a cached copy of the CICAM player speed returned in the `CICAM_player_info_reply()` APDU.

An application reading the `playbackRate` property shall result in the terminal returning the cached CICAM player speed divided by 100.

An application reading the `paused` property shall result in the terminal returning `true` if the cached CICAM player speed is zero. Otherwise the terminal returns `false`.

K.9 Random access

In the seeking procedure defined in clause 4.7.10.9 of the HTML5 specification [54], immediately following step #11, "Set the current playback position to the given new playback position", the terminal shall send a `CICAM_player_control_req()` APDU to the CICAM with the `command` field set to 0x01, the `seek_mode` field set to 0x00 and the seek position being the newly set current playback position expressed in milliseconds.

Only when the CICAM replies with a `CICAM_player_info_reply()` APDU shall the seeking procedure resume with step #12 "Wait until the user agent has established whether or not the media data for the new playback position is available, and, if it is, until it has decoded enough data to play back that position."

K.10 Tracks

`VideoTrack`, `AudioTrack` and `TextTrack` objects shall be created for content delivered by a CICAM player as defined in clause A.2.12.1 of the present document for MPEG-2 transport streams delivered to the HbbTV terminal by other mechanisms. Changes in the elementary streams delivered from the CICAM (i.e. the CICAM sending a `CICAM_player_update_req()` APDU to the host) shall result in new track objects being created based on the information in the new PMT delivered by the CICAM.

NOTE: When more than one stream of the same media type is available (e.g. audio tracks in multiple languages or for accessibility), the CICAM is expected to deliver all available streams to the HbbTV terminal.

The mechanisms for component selection defined in clause 10.2.7 of the present document shall be supported for the stream delivered from the CICAM in the same way as they would be supported for an MPEG-2 transport stream streamed over HTTP.

K.11 No mapping possible

No mapping of the following methods or properties from the HTML5 video element is possible;

- The `crossOrigin` attribute will be ignored by the CICAM player.
- The `getStartDate()` method shall return a `Date` object representing "NaN".
- The `canPlayType` method shall return the same result for a given input regardless of the presence or absence of a CICAM supporting CICAM player mode.
- HbbTV applications should not call the `addTextTrack()` method for a video element being presented by a CICAM player. Any track objects created shall have the readiness state set to 'Failed to load'.

History

Document history		
V1.1.1	June 2010	Publication
V1.2.1	November 2012	Publication
V1.3.1	October 2015	Publication