Targeted Advertising

Part 1 – Functional Requirements

Contents

Introduction

HbbTV targeted advertising enables the delivery of digital video adverts over broadband that replace part of linear TV services received by terminals via DVB-T(2)/C(2)/S(2) and perhaps IPTV when based on MPEG-2 transport streams. Thereby, digital video adverts are shown on targeted terminals instead of the original broadcast adverts, which will be seen by viewers on all non-targeted terminals. Non-commercial use cases are possible using HbbTV targeted advertising (e.g. long form non-commercial substitution content can be presented to viewers).

It is possible to deliver a targeted advertising experience using the standard features of TS 102 796 [1] – e.g. for a broadcast to broadband switch, calling the stop method on a video/broadcast object, waiting for the state change to complete and then calling play on an HTML5 video element, however the present document extends the features of TS 102 796 [1] to enable a more seamless viewing experience.

The primary features of the present document are that:

- Digital Ad Substitution (DAS) is performed under the control of an HbbTV DAS Application, i.e. DAS is not an entirely native terminal functionality
- The terminal is required to implement MSE, to enable advertising content to be pre-fetched, buffered and played under the control of the HbbTV DAS application.
- A dedicated Fast Media Switch API is defined, which allows the implementation of the content switch itself to be pushed deep down into the software stack of the HbbTV terminal, enabling the transition to be more accurate, shorter and cleaner than would be the case if it was done purely in Javascript
- The requirements in the present document can be implemented on terminals supporting only one video and one audio decoder as well as terminals supporting more than one of each. The techniques described indicate how to cater for such common differences.

The present document recognises that the capabilities, formats, and performance characteristics of broadcast and broadband streams, the DAS applications and the terminal itself cannot be precisely specified or guaranteed and variations will exist from implementation to implementation. It is possible that the terminal may well be performing other activities, including user-initiated ones, at the time the DAS is required and that this could affect DAS performance. As such a large part of the present document addresses how to mitigate between these varying capabilities as effectively as possible, adequately preparing for the necessary content switches in advance, defining windows of time during which the DAS application can perform various functions, such that the end presentation to the viewer is as high quality as possible.

For all the above reasons this document does not define precise DAS switching performance requirements, these are covered in an accompanying document, HbbTV Targeted Advertising part 2 [9].

The document also covers aspects related to usage reporting, security and privacy.

Content providers that plan to use the capabilities of a terminal implementing the present document need to be aware that such use may be subject to a (commercial) agreement between the content provider and the manufacturer of the terminal. This may be enforced by the HbbTV implementation running in the device. (see clause 7.2.1).

1 Scope

The present document extends ETSI TS 102 796 [1] to add support for the replacement of advertisements in the broadcast with ones targeted at the consumer.

2 References

2.1 Normative references

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

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

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

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

NOTE: Including the latest errata as published on http://hbbtv.org/resource-library/#specifications

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

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

[3] 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/.

[4] W3C Recommendation (17 November 2016): "Media Source Extensions".

NOTE: Available at https://www.w3.org/TR/2016/REC-media-source-20161117

[5] W3C Web Platform Incubator Community Group (Editor's Draft 30 May 2018) "Incubation of vNext 'media-source' Specification Features", codec switching

NOTE: Available at https://rawgit.com/WICG/media-source/codec-switching/index.html

[6] W3C Recommendation (26 January 2017): "Web Cryptography API "

NOTE: Available at https://www.w3.org/TR/WebCryptoAPI/

[7] ISO/IEC 13818-1:2018: "Information technology -- Generic coding of moving pictures and associated audio information -- Part 1: Systems".

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

[9] ETSI TS ??? ???–2: " Hybrid Broadcast Broadband Television: Targeted Advert Replacement; Part #2: Non-functional Requirements".

[10] W3C Candidate Recommendation (4 April 2019): "WebVTT: The Web Video Text Tracks Format"

NOTE: Available at https://www.w3.org/TR/webvtt1

[11] ETSI TS 102 034 (V1.5.1): "Digital Video Broadcasting (DVB); Transport of MPEG-2 TS Based DVB Services over IP Based Networks".

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

[13] WHATWG HTML Living Standard

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

2.2 Informative references

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

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

[i.1] Open IPTV Forum Release 2.3 specification volume 5a (V2.3): "Web Standards TV Profile".

[i.2] W3C Web Platform Incubator Community Group "Media Source Extensions: Codec Switching Explainer".

NOTE: Available at https://github.com/wicg/media-source/blob/codec-switching/codec-switching-explainer.md

[i.3] ETSI TS 103 606 "Hybrid Broadcast Broadband Television; Operator Applications".

[i.4] W3C TAG Finding (07 February 2017) "Polyfills and the evolution of the Web"

NOTE: Available at https://www.w3.org/2001/tag/doc/polyfills/

[i.5] ETSI TS 103 286-2 "Digital Video Broadcasting (DVB); Companion Screens and Streams; Part 2: Content Identification and Media Synchronization"

[i.6] Google "Inside look at modern web browser (part 1)" at https://developers.google.com/web/updates/2018/09/inside-browser-part1

[i.7] Google "Exceeding the buffering quota" at https://developers.google.com/web/updates/2017/10/quotaexceedederror

[i.8] W3C Candidate Recommendation, "Mixed Content", 2 August 2016

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

[i.9] IAB Tech Lab, "Video Ad Serving Template (VAST) - Version 4.0"

[i.10] IAB Tech Lab, "Video Ad Serving Template (VAST) - Version 4.1"

[i.11] IAB Digital Video Committee, "Digital Video In-Stream Ad Metric Definitions"

[i.12] IAB, "Digital Video In-Stream Ad Metric Definitions"

NOTE: Available at: https://www.iab.com/guidelines/digital-video-in-stream-ad-metric-definitions/.

[i.13] IETF RFC 6265 "HTTP State Management Mechanism"

[i.14] "JavaScript library for rendering IMSC Text and Image Profile documents to HTML5".

NOTE: Available at https://github.com/sandflow/imscJS

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ETSI TS 102 796 [1] and the following apply:

browser task: a task as defined in HTML5 [3]

DAS application: HbbTV application performing broadcast advert substitution as defined in the present document

landing period: a reserved time within the broadcast filled with sacrificial content during which the switch from an advert back to the broadcast can take place

sacrificial content: content within the broadcast transmission for which there are no adverse consequences if it is not seen by the TV viewer (e.g. bumper / opener / closer content)

spin the event loop: run the processes to empty the JavaScript event loop as defined in clause 7.1.4.2 of HTML5 [3]

suitable video and audio decoders: A video decoder and an audio decoder that 1) support the video and audio codec & profile for the content referenced by `newMediaObject`, 2) can be connected to the source of the content referenced by `newMediaObject` and 3) can be connected to any appropriate content protection element if the content referenced by `newMediaObject` is protected.

switch preparation deadline: The latest time before `switchTime` where the terminal would have enough time to prepare for the switch

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI TS 102 796 [1] and the following apply:

DAS	Dynamic Ad Substitution
GDPR	General Data Protection Regulation
MSE	Media Source Extensions

4 Overview

4.1 Architectures for generating DAS signalling

An HbbTV application performs Dynamic Ad Substitution ("DAS") in response to in-band or out-of-band signalling describing advert placement opportunities.

Figure 1 illustrates the information flows required from broadcaster to HbbTV application if the application is to correctly identify an advert placement opportunity in a broadcast service:

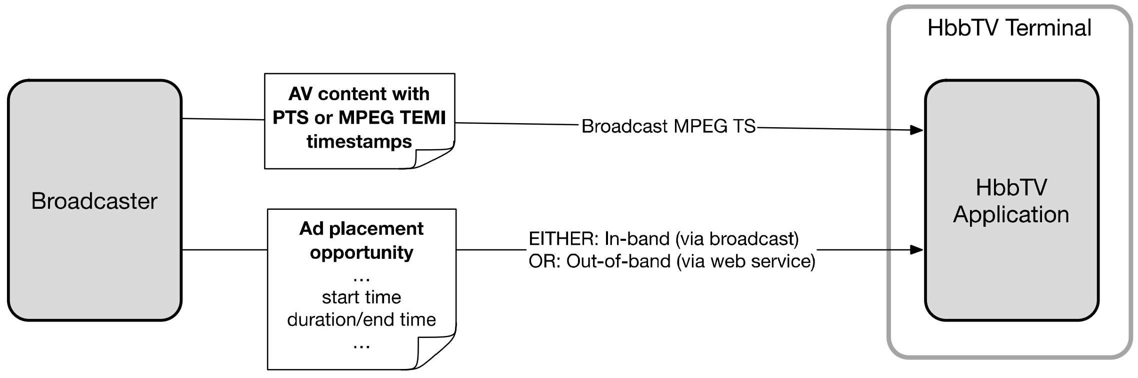


Figure 1: Information flow from broadcaster to HbbTV application for signalling an advert placement opportunity.

The signalling describing an advert placement opportunity can be delivered to the application in-band in a DSM-CC object carousel as file objects or as the payload of Do-It-Now Stream Events. It can also be delivered out-of-band as a web service.

The format of the data payload is agreed between the broadcaster and the developer of the application. The choice of delivery mechanism and the format of the data payload is an implementation choice and is therefore outside the scope of the present document.

NOTE: A broadcaster can choose to obfuscate or encrypt this data if, for example, there are concerns that it will be used to enable advert skipping functionality in broadcast receiving devices.

This data needs to be received by the HbbTV application sufficiently ahead of the substitution start time so that the application has enough time to carry out the advert decisioning process and pre-buffer the broadband delivered stream of substitution content.

The time at which the application receives this data does not need to be precise relative to the broadcast audio and video being presented. The precise timing of the start of the substitution is described in the advert placement opportunity data using timestamps denoting the start and end (or duration). These timestamps can be either PTS values relating to the timeline of the PCR clock of the broadcast service; or MPEG TEMI timestamps relating to an MPEG TEMI timeline conveyed in the broadcast service using timeline descriptors in the broadcast service as define by Annex U of ISO 13818-7 [7]. The choice of timeline is independent of how the advert placement opportunity signalling is delivered.

HbbTV terminals support receiving an MPEG TEMI timeline carried either in the MPEG transport stream packets of an existing audio or video component of the service or a separate component with `stream_type` 6 as described in clause 13.4.2 of TS 102 796 [1].

The simplest way to generate MPEG TEMI, when it is not a built-in-function of a broadcast encoder, is to do so as a separate component.

HbbTV applications performing DAS are expected to be nearly identical whether they use MPEG TEMI or PTS. A Timeline Selector (as defined in clause 13.4.2 of the HbbTV 2 specification TS 102 796 [1]) is used to specify the choice of timeline that the terminal should access from the broadcast.

4.2 Timing concepts and definitions

4.2.1 Timelines and units

A PTS timestamp carried in a broadcast transport stream refers to the presentation time of a particular access unit of media (typically a frame of the video). In the present document, the presentation time refers to the time at which that access unit starts to be presented. ISO 13818-1 [7] clause U.3.6 specifies the relationship between PTS and an MPEG TEMI timeline and how a terminal calculates an MPEG TEMI timestamp that is equivalent to a PTS timestamp (if the broadcast service carries an MPEG TEMI timeline)

Similarly, for broadband-delivered substitution content, a timestamp associated with an access unit shall refer to the time at which that access unit starts to be presented.

A timestamp representing the end of presentation of an access unit is the same value as a timestamp representing the start of presentation of the next access unit.

A PTS timestamp is encoded in units of 90 000 ths of a second. MPEG TEMI timestamps are encoded in units of ticks with a tick rate that is signalled. In the present document, all equations referencing time positions, periods, durations and timestamps are assumed to have been scaled to use the same units of time (such as seconds) unless specified otherwise.

In the present document, PTS and TEMI timestamps are assumed to be a representation of a timestamp on an infinite monotonic timeline modulo a limit M. The limit is 2^{33} ticks in the case of PTS and 2^{32} or 2^{64} ticks in the case of 32bit and 64bit MPEG TEMI time values, respectively. Therefore, any expression of an inequality relation between two timestamps X and Y shall be evaluated as follows:

```
X ≉ Y if E ≉ 0
X < Y if E < 0
X ≤ Y if E ≤ 0
X > Y if E > 0
X ≥ Y if E ≥ 0
```

Where E is calculated from X and Y as follows:

```
D = (X % M) - (Y % M)
IF D ≥ M/2 {
    E = D - M
} ELSE IF D < -M/2 {
    E = D + M
} ELSE {
    E = D
}
```

Where % is the modulus operator.

NOTE: The behaviour of the modulus operator when using a negative value varies between programming languages and in some cases is language implementation dependent. The algorithm provided here behaves correctly for negative values of X and/or Y for all implementations of the modulus operator.

Determining whether a timestamp lies in the past or future on a given timeline shall be evaluated in the same way using an equivalent inequality relation between that timestamp and another representing the timeline position corresponding to the video frame currently being composed with the application's graphics by the terminal.

4.2.2 Advance timing of API calls by the application

If the terminal requires that an API call is made by the application at least *N* seconds in advance of a timestamp *T* then it needs to take place at or before timestamp *S* where:

$$S = T - N$$

In the present document, a single spin of the JavaScript event loop is considered to take place at timestamp *S* if any changes it makes to the DOM result in graphics that are composed with the video frame with timestamp S.

NOTE 1: An application can calculate an estimate of the current timestamp by reading one of the `currentTime` properties described in clause 13.11 of TS 102 796 [1] and subtracting 100ms and the duration of one video frame. This is to take into account the limits on reading accuracy and the fact that the reading refers to the previously composed frame.

NOTE 2:` Timing of execution of JavaScript is best effort only and additional delays could be incurred depending on the complexity of the code. Application developers are strongly advised to leave additional safety margin when timing the execution of an API call.

EXAMPLE: A terminal is presenting broadcast video with a frame duration of 0.04 seconds (25 frames per second). An application intends to call an API function to request an action take place at PTS timestamp 10.0 seconds (PTS value of 900 000 ticks). The terminal requires this action be scheduled at least 1.0 second in advance. The application queries the `currentTime` property of a `MediaSynchroniser` initialised with a PTS timeline selector. If the value is less than 8.86 then an immediate call to the API function will be sufficiently in advance for the terminal to act on it.

If the terminal defines this requirement in relation to the broadcast timeline while broadband content is being presented then in this situation the timings should be calculated as if broadcast were still being presented.

The threshold described by timestamp S is with respect to the same timeline as is used for timestamp T. Some terminal implementations may allow timestamp T to be defined with respect to a broadcast timeline, even when broadband substitution content is being presented. In these circumstances, the time at which the threshold is reached 1 be calculated with respect to the broadcast timeline and not to the timeline of the broadband substitution content being presented.

4.2.3 Broadcast to broadband switch: accuracy

When an application requests a switch from broadcast media to broadband media, the switch is scheduled at a media time T1 ("the scheduled switch time") which is the value provided by the application, rounded up to align to a broadcast video frame boundary.

A terminal that is capable of switching from broadcast to broadband within an accuracy limit of A1 seconds shall be observed to begin the switch at a time no earlier than the scheduled switch time T1 and no later than T1+A1. The accuracy limit A1 shall be a non-negative value and represents the maximum amount by which the switch shall be observed to begin after the scheduled switch time.

A1 is not defined by the present document. HbbTV Targeted Advertising part 2 [9] defines performance profiles including a value for A1.

The time at which the switch is observed to begin shall be determined as when either broadcast video or broadcast audio ceases presentation for the end user, whichever happens first. The time at which broadcast video ceases presentation can be different to the time at which broadcast audio ceases presentation. However, both shall have ceased before T1+A1 and before presentation of broadband audio or video is observed to begin at the end of the switch period.

Broadcast video presentation shall only end aligned to a frame boundary and shall not end mid-way through the presentation of a frame.

More formally, a terminal that is capable of switching from broadcast to broadband within an accuracy limit of A1 seconds shall begin a switch that is scheduled to begin at timestamp T1 at any time X where:

- X is the earliest of X_V and X_A where
  - X_V is the time at which presentation of broadcast video ceases for the end-user and
  - X_A is the time at which presentation of broadcast audio ceases for the end-user.

The time X_V at which presentation of broadcast video ceases shall be such that:

- X_V is aligned to the time of start of presentation of a broadcast video frame; and
- $T1 \le X_V \le T1 + A1$.

Ceasing broadcast video presentation at time X_V on the timeline of the broadcast means that broadcast video is presented up to the video frame immediately before the one with timestamp X_V. Broadcast video frames with timestamp X_V (and later) are not presented.

The time X_A at which presentation of broadcast audio ceases shall be such that:

- $T1 \le X_A \le T1 + A1$.

Ceasing broadcast audio presentation at time X_A on the timeline of the broadcast means that broadcast audio is presented up to that time, but from time X_A onwards, no broadcast audio is being presented for the end-user.

Given a switch period with duration D (as defined in clause **4.2.4**) after the switch is observed to begin, X_V and X_A shall also be such that:

- $X_V \le \mathrm{X} + \mathrm{D}$ and
- $X_A \le \mathrm{X} + \mathrm{D}$.

EXAMPLE 1: The terminal is presenting broadcast video at 25 frames per second. The video frames have PTS timestamps starting at 0 and incrementing by 3 600 per video frame. A switch is requested by the application at PTS timestamp 87 000. The terminal is capable of switching within an accuracy limit of 150 milliseconds (13 500 PTS ticks) and a switch duration not exceeding 100 milliseconds.

The terminal rounds the switch time up to 90 000 to align with a video frame boundary. The terminal will therefore be observed to cease presentation of broadcast video on a frame whose PTS timestamp is greater than or equal to 90 000 and less than or equal to 103 500. The possible PTS timestamps, aligned to the start of presentation of frames, at which the presentation of broadcast video could cease are therefore 90 000, 93 600, 97 200 and 100 800. This is illustrated in Figure 2. Broadcast audio presentation could cease at any time between the first of those frames and 150 milliseconds after that.

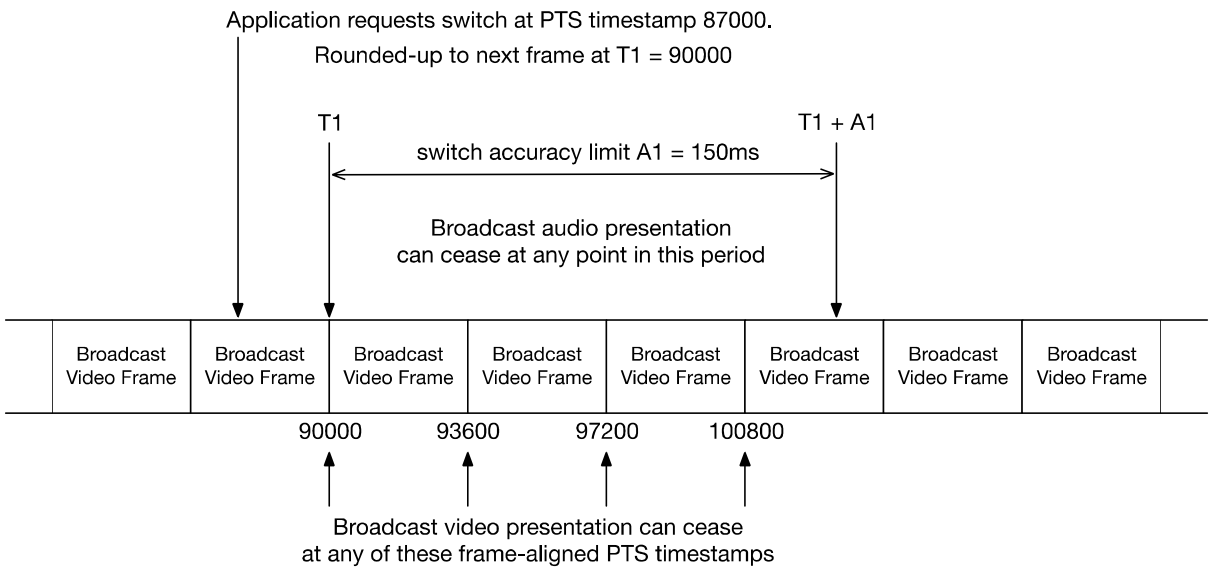

Application requests switch at PTS timestamp 87000.
Rounded-up to next frame at T1 = 90000
T1
T1 + A1
switch accuracy limit A1 = 150ms
Broadcast audio presentation can cease at any point in this period
Broadcast Video Frame
90000
93600
97200
100800
Broadcast video presentation can cease at any of these frame-aligned PTS timestamps

Figure 2: Example showing the range of possible times at which a broadcast to broadband switch can begin given a stated accuracy limit

EXAMPLE 2: In one scenario based on example 1, the terminal is observed to cease presentation of broadcast video before it ceases presentation of broadcast audio. It does so aligned to a video frame boundary at PTS timestamp 90 000. This means that the final frame of broadcast video that is presented has PTS timestamp 86 400. The switching period is observed to have begun at the moment the terminal would have begun presenting the video frame with PTS timestamp 90 000. Presentation of broadcast audio needs to cease before T1+A1 and also before presentation of broadband audio or video begins. Therefore it is constrained to cease no later than 100 milliseconds after the beginning of the switch. This is illustrated in figure 3.

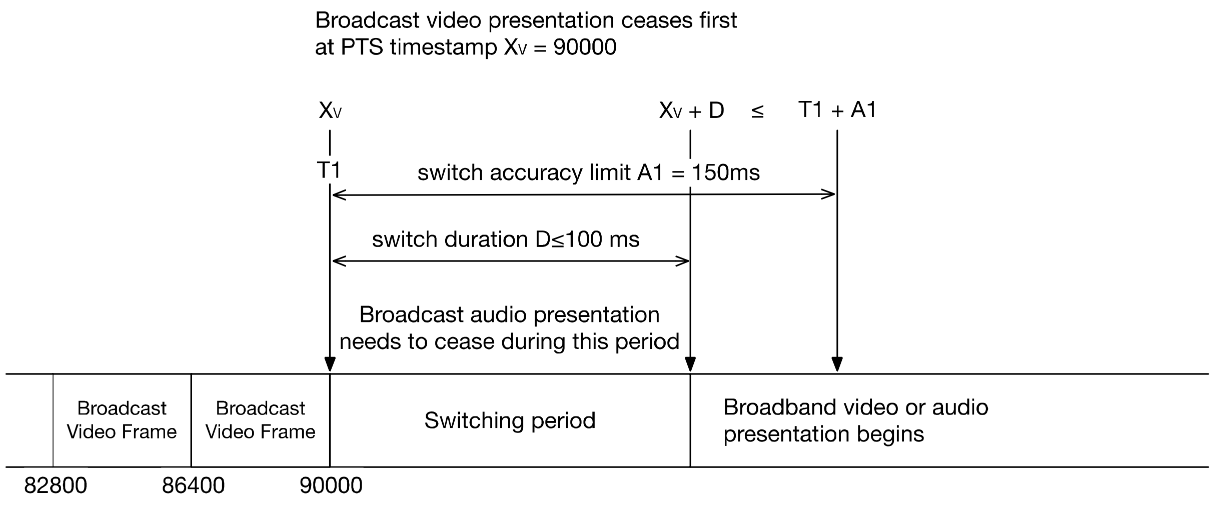

Broadcast video presentation ceases first at PTS timestamp Xv = 90000
Xv
Xv + D ≤ T1 + A1
T1
switch accuracy limit A1 = 150ms
switch duration D≤100 ms
Broadcast audio presentation needs to cease during this period
Broadcast Video Frame
Broadcast Video Frame
Switching period
Broadband video or audio presentation begins
82800
86400
90000

Figure 3: Example showing broadcast to broadband switch where presentation of video ceases first and switch duration constrains how much later presentation of broadcast audio can cease

EXAMPLE 3: In another scenario based on example 1, the terminal is observed to cease presentation of broadcast audio before it ceases presentation of broadcast video. It does so at a time equivalent to PTS timestamp 96 500. Broadcast video ceases at a later point that is aligned to a video frame boundary and needs to be both before time T1+A1 and before the presentation of broadband video or audio begins. Broadcast video is therefore constrained to cease at either PTS timestamp 97 200 (if the switch period duration is at least 7.8 milliseconds) or at PTS timestamp 100 800 (if the switch period duration is at least 47.8 milliseconds). This is illustrated in figure 4.

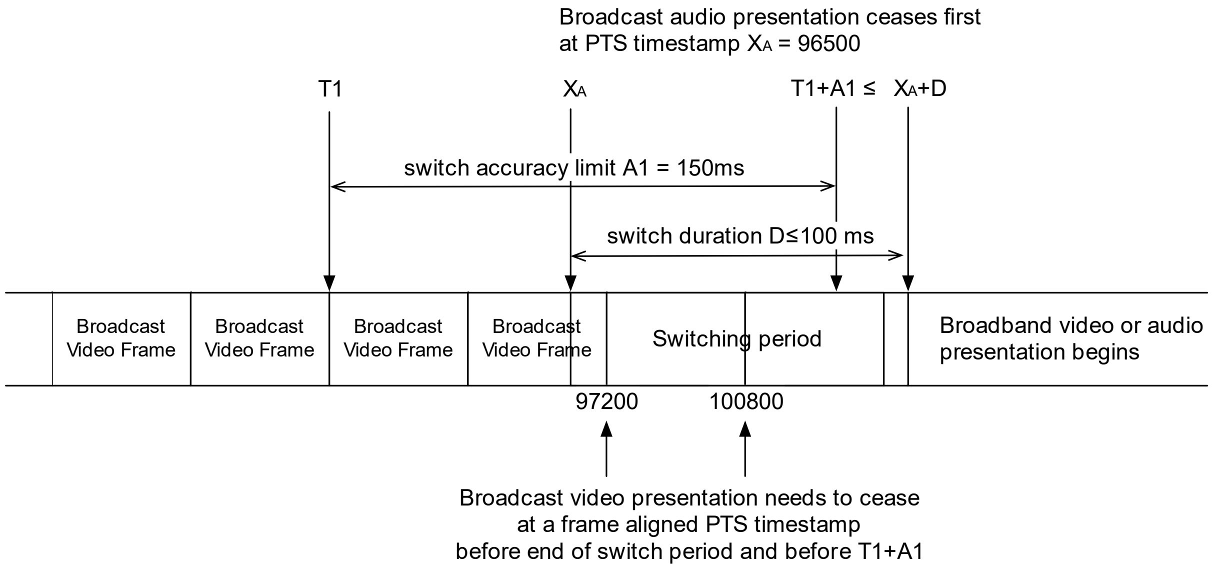

Broadcast audio presentation ceases first
at PTS timestamp XA = 96500
T1
XA
T1+A1 ≤ XA+D
switch accuracy limit A1 = 150ms
switch duration D≤100 ms
Broadcast Video Frame
Broadcast Video Frame
Broadcast Video Frame
Broadcast Video Frame
Switching period
Broadband video or audio presentation begins
97200
100800
Broadcast video presentation needs to cease
at a frame aligned PTS timestamp
before end of switch period and before T1+A1

Figure 4: Example showing broadcast to broadband switch where presentation of audio ceases first and switch accuracy constrains how much later presentation of broadcast video can cease

4.2.4 Broadcast to broadband switch: duration

The switch period for the switch from broadcast to broadband is the time period between the beginning of the switch (as defined in clause 4.2.3) and the beginning of presentation of the broadband video or audio, as observed by the end-user.

NOTE: The observed beginning of the switch is when either broadcast video or broadcast audio is observed by the end-user to cease presentation, whichever happens first.

When a broadcast to broadband switch takes places, a terminal that is capable of a broadcast to broadband switch with minimum duration D1min and maximum duration D1max shall be observed by the end-user to have a switching period of duration D1 where $D1min \leq D1 \leq D1max$.

D1min and D1max are not defined by the present document. HbbTV Targeted Advertising part 2 [9] defines performance profiles including values of D1min and D1max.

4.2.5 Broadband to broadcast switch: accuracy

Broadband media can comprise any combination of video, audio or subtitle (caption) components. When an application requests a switch from broadband media back to broadcast media, the switch is scheduled at a media time T2 (the "scheduled switch time") as follows:

- If the application does not explicitly provide a switch time, or the value provided is after the end time of the broadband media, then T2 shall be the timestamp corresponding to the end time of the broadband media.
  - If the media resource for the broadband media has a defined duration, then the end time shall correspond to its start plus that duration;
  - otherwise, the end time shall correspond to the last media time for which there is data for any of the media components.
- Otherwise, if the value provided by the application corresponds to a time before the end of the video

component, then T2 shall be the value provided by the application, rounded up to align to a broadband video frame boundary.

- Otherwise, T2 shall be the value provided by the application.

NOTE 1: Individual components can be shorter than the overall duration of the broadband media. Some components can therefore have already ended prior to the scheduled switch time T2.

A terminal may support allowing the application to specify a switch timestamp on a timeline of the broadcast. When this is the case, the terminal shall interpret that timestamp as if broadcast is still being presented.

NOTE 2: For a terminal to be able to support an application specifying the timing of a broadband to broadcast switch in terms of a broadcast timeline, the terminal might need to continue to decode (without presenting) the broadcast video and/or audio while it is presenting a broadband stream.

A terminal that is capable of switching from broadband to broadcast within an accuracy limit of A2 seconds shall be observed to begin a switch no earlier than the scheduled switch time T2 and no later than T2+A2. The accuracy limit A2 shall be a non-negative value and represents the maximum amount by which the switch shall be observed to begin after the scheduled switch time.

A2 is not defined by the present document. HbbTV Targeted Advertising part 2 [9] defines performance profiles including a value for A2.

The time at which the switch is observed to begin shall be determined as follows, depending on whether the scheduled switch time is past the end of broadband video and/or audio:

- If the scheduled switch time is past the end of both the broadband video and audio then the time at which the switch is observed to begin is outside the scope of the present document;
- Otherwise if the scheduled switch time is past the end of the broadband video but not past the end of the broadband audio, then the time at which the switch is observed to begin shall be when broadband audio ceases presentation to the end-user;
- otherwise, if the scheduled switch time is past the end of the broadband audio but not past the end of the broadband video, then the time at which the switch is observed to begin shall be when broadband video ceases presentation to the end-user;
- otherwise, the time at which the switch is observed to begin shall be when either the broadband video or the broadband audio ceases presentation, whichever happens first.

NOTE 3: When the presentation of subtitles ceases is intentionally ignored for determining the time at which the switch begins.

Even when the scheduled switch time is at or before the end of both the broadband video and the broadband audio, the time at which broadband video ceases presentation can be different to the time at which broadband audio ceases presentation for the end user. However in all cases, both shall have ceased before T2+A2 and before presentation of broadcast audio or video is observed to begin at the end of the switch period.

Broadband video presentation shall only end aligned to a frame boundary and shall not end mid-way through the presentation of a frame.

More formally, a terminal that is capable of switching from broadband to broadcast within an accuracy limit of A2 seconds shall be observed to begin a switch that is scheduled for timestamp T2 at any time Y where:

- If $E_V \leq T2$ then Y is Y_A else if $E_A \leq T2$ then Y is Y_V else Y is the earliest of Y_V and Y_A where
- Y_V is the time at which presentation of broadband video ceases for the end-user and
- Y_A is the time at which presentation of broadband audio ceases for the end-user and.
- E_V and E_A are the times at which presentation of broadband video and audio (respectively) would cease for the end-user if the broadband media resource were played to its end.

If $E_V > T2$ then the time Y_V at which presentation of broadband video ceases shall be such that:

- $T2 \leq Y_V \leq T2 + A2$; and

- Y_V is either:
  - aligned to the time of start of presentation of a broadband video frame; or
  - at or after the end of presentation of the final frame of the broadband video

NOTE 4: If Yv is after the end of the first presentation of the final frame of the broadband video, HTML5 [3] requires that a video element that is played to the end becomes paused and that the video element continues to represent the last frame of video to have been rendered.

Ceasing broadband video presentation at time Y_V on the timeline of the broadband media means that broadband video is presented up to the video frame immediately before the one with timestamp Y_V. Video frames with timestamp Y_V (and later) are not presented.

If $E_A > T2$ then the time Y_A at which presentation of broadband audio ceases shall be such that:

- $T2 \leq Y_A \leq T2 + A2$.

Ceasing broadband audio presentation at time Y_A on the timeline of the broadband means that broadband audio is presented up to that time, but from time Y_A onwards, no broadband audio is being presented for the end user.

Given a switch period with duration D2 (as defined in clause 4.2.6) after the switch is observed to begin, Y_V and Y_A shall also be such that:

- either $(E_V < T2)$ or $(Y_V - \mathrm{Y} \leq \mathrm{D2})$ and
- either $(E_A < T2)$ or $(Y_A - \mathrm{Y} \leq \mathrm{D2})$.

4.2.6 Broadband to broadcast switch: duration

The switch period for the switch from broadband to broadcast is the time period between when the switch begins (as defined in clause 4.2.5) and the beginning of presentation of the broadcast video or audio, as observed by the end-user.

A random access point for broadcast video is a coded frame that can be decoded without reference to past or future frames. The first frame in a coded Group of Pictures (GOP) is typically a random access point. The number of frames in a Group of Pictures is the GOP length. Let N be the GOP length of the broadcast during the period in which the switch could commence (starting at timestamp T2 and ending at timestamp T2+A2, as defined in clause 4.2.5). Let F be the frame rate in frames per second.

EXAMPLE: If N is 1 and F is 25 then every frame is a random access point and the GOP length is 1 frame.

If N is 5 and F is 50 then the GOP length is 5 frames.

A terminal that is capable of a switch from broadband to broadcast shall be capable of doing so with duration D2, as observed by the end-user, where:

- $\mathrm{D2min} \leq \mathrm{D2} \leq D2max$ if the terminal is capable of continuing to decode broadcast video and audio while presenting broadband video and audio, or if the GOP length is 1;
- otherwise $D2\mathrm{min} \leq \mathrm{D2} \leq D2max + \left(\frac{N-1}{F}\right)$

where:

- D2min is the minimum broadband to broadcast switch duration; and
- D2max is the maximum broadband to broadcast switch duration when the GOP length N is 1 frame; and
- D2, D2min and D2max are in units of seconds.

D2min and D2max are not defined by the present document.HbbTV Targeted Advertising part 2 [] defines performance profiles including values of D2min and D2max.

EXAMPLE 2: D2min is 40 milliseconds and D2max is 250 milliseconds The broadcast video GOP length N is 100 frames and the frame rate F is 50 fps.

(N-1)/F is 1 980 milliseconds and represents the difference in duration between a GOP of length 100 and a GOP of length 1.

The minimum duration of the broadband to broadcast switch is 40 milliseconds.

If the terminal is capable of continuing to decode broadcast while presenting broadband then the maximum duration of the switch is 250 milliseconds, otherwise the maximum duration of the switch is 250 + 1 980 = 2 230 milliseconds.

4.2.7 Drift

When PCR and PTS and the playback rate of broadcast audio and video are running within a tolerance of ±30ppm (as specified by ISO 13818-1 [7]) then a terminal with a maximum drift rate δ ppm plays broadband audio and video such that, on average, every second of broadband content playback varies in duration compared to a second of broadcast content by an amount not exceeding ± (30+δ) microseconds.

EXAMPLE: A terminal has a maximum drift rate δ of 333 ppm. Between 119.96 and 120.04 seconds of broadcast video (2 minutes ±1 frame at 25fps) will have elapsed after 120 seconds of broadband video playback.

NOTE: Drift can be zero if the terminal maintains synchronisation between broadcast and broadband during broadband presentation (because, for example, it continues to decode broadcast while also decoding and presenting broadband).

5 User experience (informative)

5.1 Introduction

This clause describes the behaviour of the terminal as seen by the end-user. It should be considered as usability guidelines for implementing Dynamic Ad Substitution. The described behaviour results from a combination of:

- the timing and content of the broadcast content;
- the functionality coded into the broadcast application;
- and the capabilities of the terminal and its switching performance in terms of achievable accuracy and duration of switch.

A consistent and smooth user experience is an important factor in enabling a user experience that is acceptable to the end-user. To ensure this, both the manufacturer and the application developer should respect the following framework and guidelines.

5.2 Timings for take-off and landing periods in the broadcast content

Broadcasters can prepare the broadcast for dynamic ad substitution by including a ‘take-off period’ and a 'landing period' of 'sacrificial content' to allow for varying durations and accuracies in switching from the broadcast to the ad and back again. Examples of sacrificial content include plain black or the broadcaster logo.

Broadcasters wishing to avoid switches that result in single frame "flashes" of broadcast content can consider the following.

- For the ‘take off period’, e.g.:
  - Using black or a freeze-frame of the end of the preceding content as the sacrificial content; and/or

  - Lengthening the 'take-off period' and scheduling the switch to start a several frames after the beginning of the 'take-off period'.

- For the 'landing period', e.g.:
  - Using black as the sacrificial content; and/or
  - Lengthening the 'landing period' such that the slowest switch back to broadcast will end several frames before the 'landing period' ends.

If a broadcaster wants to support HbbTV terminals that can only decode one stream at one time then such a landing period would ideally be specially encoded in the broadcast to have a higher than average proportion of I-frames (or equivalent).

NOTE 1: At the time of writing there is no known professional broadcast equipment that supports dynamically adjusting the GOP length.

Other factors determining the length of a landing period include the following;

- Some existing broadcast production equipment may have configurations that are impractical to change for the introduction of targeted advertising. For example, automatic insertion by the encoder of a certain number of frames at each content change (content-advert // advert-advert // advert-content).

NOTE 2: If an encoder would insert 6 frames at each content change then this would mean 240 ms between each advert and 240 ms between the last advert of a break and the content afterwards.

NOTE 3: When the landing period is in the middle of an ad break, missing the end of the landing period will cause commercial issues with the owner of the next advert.

- If implementations that only support decoding one video stream at one time and one audio stream at one time are to be supported, the landing zone may need to be as large as the sum of the GOP interval and two times the largest switching duration and worst accuracy to be supported.

NOTE 4: For example, supporting terminals with switching duration upto 240ms, switching accuracy upto 40ms with a GOP duration of 1.5s would require landing periods of just over 2s.

- Longer landing periods are believed to be more easily accommodated at the end of an advert break. This is for both commercial reasons as well as the tolerance of consumers who watch the original (non-substituted) broadcast.

Annex C.2 describes a model for determining the timing requirements of take-off and landing periods in a broadcast.

5.3 Visibility and z-axis ordering of DOM elements

This clause describes the user experience in the simplest scenario where a single substitution takes place. The substitution content consists of a sequence of one or more adverts assembled as a single stream. Other scenarios are supported, including ones where multiple separate substitutions occur consecutively with no return to broadcast between them.

The HbbTV application is responsible for creating the DOM elements involved in the process of Dynamic Ad Substitution. The HbbTV application is also responsible for ensuring the elements are in an appropriate state prior to the switch from broadcast to broadband and back to broadcast again. Both switches are performed by the terminal at the request of the application and the terminal is responsible for making the necessary changes to the DOM during this process.

The application is also in control of what is displayed on screen during the periods in which the switch and switch back are taking place. The application can, for example, choose to display a full screen graphic during longer duration switching periods or black during short ones that last only a few frames. Figure 5 illustrates what the user sees during this process and the visibility of relevant elements within the DOM:

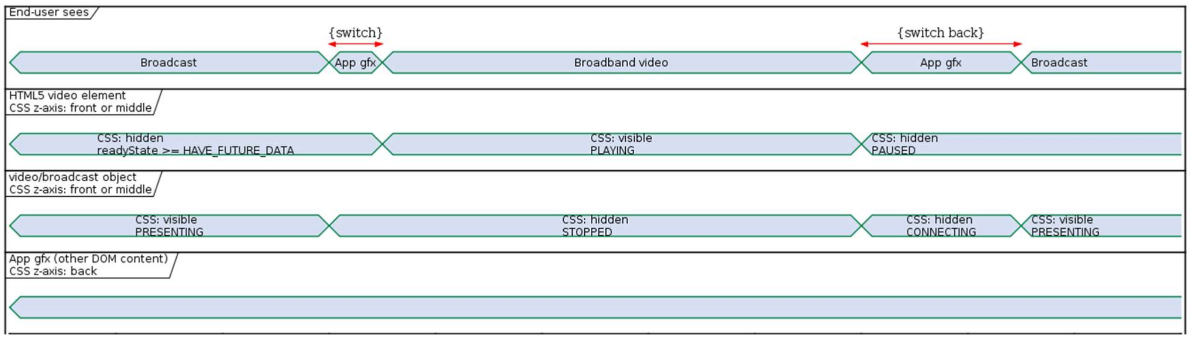


Figure 5: Timing diagram showing how DOM element visibility determines what the end-user sees during the Dynamic Ad Substitution process

Initially, the terminal is presenting broadcast video. The application has instantiated a bound video/broadcast object that is in the `PRESENTING` state. In preparation for the switch from broadcast to broadband, the application has created an HTML5 video element with its CSS `visibility` property set to `hidden` and has placed it either in-front or behind the video/broadcast object in the CSS z-axis. The HTML5 video element will be used to play a broadband stream. This is the content that will be substituted in place of broadcast content.

The application uses the API (defined in clause 8.1 of the present document) to instruct the terminal to perform the switch. The switch begins when the terminal stops broadcast presentation, resulting in the video/broadcast object transitioning to the `STOPPED` state, and the terminal sets its CSS `visibility` property to `hidden`. This immediately reveals to the end-user any other application defined content in the DOM behind the video/broadcast object and HTML5 video element elements in the CSS z-axis.

To complete the switch, the terminal also commences playback of the HTML5 video element. When it begins to play, the terminal immediately sets its CSS `visibility` property to `visible` ensuring that the first frame is displayed.

The switch back is also initiated via the same API. The terminal begins the switch back either at the end of the broadband stream or at a specified time index at which the terminal pauses the HTML5 video element. The terminal also immediately sets its CSS `visibility` property to `hidden`. Again, this reveals to the end-user any application defined DOM content that is behind these two elements in the CSS z-axis.

To complete the switch back, the terminal also begins transitioning the video/broadcast object back to the `PRESENTING` state. When it reaches the `PRESENTING` state, the terminal immediately sets its CSS `visibility` property to `visible`.

The terminal ensures that changes in CSS `visibility` properties are timed such that the video/broadcast object and HTML5 media elements are only visible when they are showing video content and are not visible at other times. This avoids, for example, showing frames of black to the end-user during both switch and switch back when the application intends to show some other DOM content such as a full screen graphic.

6 Service and application model

6.1 Introduction

For regular HbbTV applications, the service and application model shall be identical to that specified in TS 102 796 [] except as specified here. For operator applications, the service and application model shall be identical to that specified in TS 103 606 [i.3].

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

The requirements in clause 6.2.2.7 of TS 102 796 [1] defining that “Terminals shall be able to present broadband delivered video at the same time as demultiplexing MPEG-2 sections from the broadcast" shall also apply for video presented via MSE at the same time as demultiplexing MPEG-2 sections from the broadcast - including all the listed examples.

7 Formats and protocols

7.1 General

Except as stated below, the present document does not introduce any additional requirements beyond those in TS 102 796 [1].

7.2 Broadcast specific formats and protocols

7.2.1 Signalling of applications

In the transport protocol descriptor (see clause 7.2.3.1 of TS 102 796 [1] and clause 5.3.6.2 of TS 102 809 [8]), the contents of the `URL_base` field (i.e. the initial page of the application) shall be traceable back to the broadcaster of the service.

7.3 Broadband-specific format and protocols

7.3.1 Subtitles

MSE [4] only requires support for `SourceBuffer`s for video and audio. There is no requirement for `SourceBuffer`s for subtitle data to be supported. Display of subtitles with video and audio presented using MSE needs to be done either by JavaScript code in the application (e.g. using `WebVTTCue` (see clause A.3.4) as used by imscJS [i.14]) or as out of band subtitles as defined in clause A.2.12.2 of TS 102 796 [1].

8 Browser application environment

8.1 Fast Media Switch API

8.1.1 Introduction (informative)

The Fast Media Switch API enables an application to define ahead of time when a switch is to be made from media presented by one object or element to media presented by another object or element. The application can define the time at which the switch is to be made with reference to a timeline associated with either the original media or the destination media. Using a dedicated API for this allows the implementation of the switch itself to be pushed deep down into the software stack of the HbbTV terminal which should enable the transition to be more accurate, shorter and cleaner than would be the case if it was done purely in JavaScript. It should also be possible to implement the Fast Media Switch API in JavaScript without those benefits (e.g. as a web "polyfill" [i.4]) to enable common application code to also be used where the Fast Media Switch API is not supported.

The design of the Fast Media Switch API is intended to support a range of possible implementations including optimised ones and non-optimised ones and allow the benefits of optimised implementations to be exploited. It is intended to be independent of the number of available (hardware) media decoders.

8.1.2 Object

The terminal shall support a non-visual embedded object of type "`application/hbbtvMediaSwitcher`" and class `MediaSwitcher` with the JavaScript API defined below.

8.1.3 Method signatures

`Promise switchMediaPresentation( Element originalMediaObject, String timelineSelector, Boolean timelineSource, Number switchTime, Element newMediaObject, String minimumSwitchPerformanceRequired)`		
Description	Requests a switch from the media being presented by `originalMediaObject` to the media to be presented by `newMediaObject`. The switch is to take place at time `switchTime` as defined in the timeline identified by `timelineSelector` carried in the media identified by `timelineSource`.	
Arguments	`originalMediaObject`	Either a video/broadcast object or an HTML5 video element.
	`timelineSelector`	Either a URN identifying a timeline or null indicating that the switch is to happen when the end of the content being played by `originalMediaObject` is reached.
	`timelineSource`	If true indicates that the timeline is carried in the `originalMediaObject`, if false indicates that the timeline is carried in the `newMediaObject`.
	`switchTime`	If `timelineSelector` is a URN then this defines the time in seconds at which to switch from `originalMediaObject` to `newMediaObject` on the timeline defined by `timelineSelector` and `timelineSource`. If `timelineSelector` is null then this is irrelevant.
	`newMediaObject`	Either a video/broadcast object or an HTML5 video element.
	`minimumSwitchPerformanceRequired`	Identifies zero or one minimum performance profiles that apply to the switch. If no performance profiles apply then this argument shall be an empty String. The requested switch shall proceed even if it is known to the terminal that either the switch duration or the switch accuracy or both would be worse than those required for all supported performance profiles. (For example, the switch is a non-time-critical switch to local news). If a performance profile applies to the switch then this argument shall be a URN that is included in a `profile` element in the `ta` element in the XML capabilities (see clause 10.3.1). The requested switch shall fail if it is known to the terminal that the performance requirements for the specified profile will not be met. Even when `minimumSwitchPerformanceRequired` identifies one or more performance profiles that are supported by the terminal and the switch does not fail, there is a small probability that the switch may not meet the timing requirements due to real-time effects that cannot be predicted in advance. See clause 9.3.4 of the present document.
Returns	`Promise`	The method always returns a Promise and never raises a synchronous exception.

8.1.4 Algorithms

8.1.4.1 When `switchMediaPresentation` is called

When this method is called, the terminal shall run the following steps.

1) Let `promise` be a new promise

2) If any of the following preconditions are not met at the time of the call to the method then return a promise rejected with a `NotSupportedError`.

   - `originalMediaObject` is either a video/broadcast object or an HTML5 video element.

- If `originalMediaObject` is a video/broadcast object then all of the following are met:
  - `originalMediaObject` is in the presenting state; and
  - `newMediaObject` is an HTML5 video element.
- If `originalMediaObject` is an HTML5 video element then all of the following are met:
  - `newMediaObject` is either an HTML5 video element or a video/broadcast object; and.
  - the `readyState` of the `originalMediaObject` is either HAVE_FUTURE_DATA or HAVE_ENOUGH_DATA.
- If `newMediaObject` is a video/broadcast object then all of the following are met:
  - its CSS visibility is set to `hidden` ; and
  - it is in either the stopped or the presenting state.

NOTE 1: If `newMediaObject` is an HTML5 video element then either it needs to have CSS visibility set to `hidden` or the application needs to take care to ensure that the video element is transparent black and not showing either a poster or a video frame.

- If `newMediaObject` is an HTML5 video element then the `readyState` is HAVE_ENOUGH_DATA.
- If `newMediaObject` is an HTML5 video element then the `seeking` attribute is `false`.
- If `timelineSource` is `true` then the timeline indicated by `timelineSelector` either
  - is `null` or
  - indicates a timeline that is supported for `originalMediaObject`

NOTE 2: See clause 10.2 for a definition of which timelines are required to be supported under what circumstances.

- If `timelineSource` is `false` then the timeline indicated by `timelineSelector` indicates a timeline that is supported for `newMediaObject`.

NOTE 3: See clause 10.2 for a definition of which timelines are required to be supported under what circumstances.

NOTE 4: `timelineSource` can be `false` regardless of the state of the video/broadcast object since, on an implementation that can decode more than one stream, decoding of the broadcast will either 1) be kept running through the advert break or 2) be started as soon as the app calls `switchMediaPresentation` for the switch back to the broadcast after the advert break - which should be immediately at the start of the last advert.

- If `timelineSelector` is `null` then `originalMediaObject` is an HTML5 video element.

NOTE 5: This indicates that the switch is to take place at the end of the media referenced in `originalMediaObject`

- `originalMediaObject` and `newMediaObject` shall have the same parent element in the DOM.
- Either `originalMediaObject` is immediately in front of `newMediaObject` on the CSS z-axis or `originalMediaObject` is immediately behind `newMediaObject` on the CSS z-axis and the latter has its CSS visibility set to `hidden`.
- `minimumSwitchPerformanceRequired` shall either be an empty string or one of the URNs included in a `profile` element in the `ta` element (see clause 10.3.1).

3) If a previous call to `switchMediaPresentation` has not yet completed then return a promise resolved with "CallInProgress".

4) If `timelineSelector` is `null` then return `promise`.

5) If not already monitoring it, start asynchronously monitoring the timeline indicated by `timelineSelector` from `originalMediaObject` or `newMediaObject` as defined by `timelineSource`. If `timelineSource` is `true`

then, except for TEMI, the terminal will be monitoring the timeline as part of decoding the media concerned. For TEMI, if the application has previously called `MediaSynchroniser.initMediaSynchroniser` with `timelineSelector` then it will be monitoring that timeline.

6) If `timelineSelector` is not null and if the time indicated by `switchTime` is determined (using the algorithm in clause 4.2.1) in the past or more than 10 minutes into the future as defined by the timeline then return a promise resolved with "InThePast".

7) If current time is after the switch preparation deadline then return a promise resolved with "SwitchPreparationDeadlinePassed".

8) Asynchronously invoke the algorithm for attempting to allocate suitable video and audio decoders for `newMediaObject`.

9) Return `promise`.

8.1.4.2 Between a call to `switchMediaPresentation` and the start of a switch

If `timelineSelector` specifies a TEMI timeline that is not already being monitored then, if the current timeline time is not known to the terminal within 2.5 seconds of starting to monitor the timeline in the `switchMediaPresentation`() call, then the requested switch shall be aborted and promise shall be rejected with a `NotSupportedError`.

For PTS and all other timelines, if the current timeline time is not known to the terminal promptly after starting to monitor the timeline in the `switchMediaPresentation`() call, then the requested switch shall be aborted and promise shall be rejected with a `NotSupportedError`.

Once the terminal first retrieves the timeline time or if there is a discontinuity in the monitored timeline, then:

- If the time indicated by `switchTime` is in the past as defined by the timeline then the requested switch shall be aborted and promise shall be rejected with an `InvalidStateError`.
- If the current time is after the switch preparation deadline then the requested switch shall be aborted and promise shall be rejected with an `InvalidStateError`.

NOTE 1: See clause 4.2.2 for an explanation of the concept of this method being required to be called sufficiently in advance of the switching time. See clause 10.2.4 for the requirements on the minimum advance notice for a switch.

- If the switch time is more than ten minutes in the future then the requested switch shall be aborted and promise shall be rejected with an `InvalidStateError`.

If monitored timeline becomes unavailable, then the requested switch may be aborted. In this case the promise shall be rejected with a `NotSupportedError`.

NOTE 2: In the present document, only the TEMI timeline can become unavailable – if it stops being signalled.

The following checks (and any resulting actions) shall be made by the terminal between when a call to `switchMediaPresentation` returns and when either `promise` is rejected or the executing the switch algorithm starts (see clause 8.1.4.3).

1) If `originalMediaObject` is a video/broadcast object and a channel change succeeds (regardless of how this channel change was initiated) then abort the requested switch and resolve `promise` with "ChannelChanged". This happens before the `ChannelChangeSucceededEvent` is posted.

NOTE 1: This may be overtaken by the application being killed if it is not permitted to run in the new channel.

2) If `originalMediaObject` is an HTML5 video element and the src attribute is changed then abort the requested switch and resolve `promise` with "SourceChanged".

NOTE 2: HTML5 defines that "Dynamically modifying a source element and its attribute when the element is already inserted in a video or audio element will have no effect."

3) If `newMediaObject` is an HTML5 video element and it changes state or behaviour as a result of the application setting any property or calling any method on `newMediaObject` then resolve `promise` with "NewObjectChanged". This may happen any time up to when the fast media switch algorithm starts.

Examples include calling `play` or `pause` or setting `currentTime` or changing the value of the `src` attribute (if used) or changing the value of the `source` element (if used) or changing `AudioTrack.enabled` or `VideoTrack.selected` on one or more of the embedded video or audio tracks of the media resource.

4) If the user makes a change using the terminal's audio language or description (or other) selection mechanism (see clause 10.2.7 of TS 102 796 [1]) then:

- If the content referenced by `newMediaObject` has only one audio track then present that audio track after the switch. i.e. effectively the change made by the user will be ignored while the ad is playing.

NOTE 3: It is strongly recommended that audio language and accessibility be handled by the application passing the corresponding preferences to the primary ad server and obtaining an advert with one audio track that matches the user preferences at that point.

- Otherwise if `originalMediaObject` is a video/broadcast object then abort the requested switch and resolve `promise` with "AudioTrackChanged".

- Otherwise if `originalMediaObject` is an HTML5 video element and `newMediaObject` is an HTML5 video element then proceed with the switch but no timing requirements apply to the switch.

NOTE 4: Under these circumstances, applications need to take care to avoid missing the landing period on a later switch back to broadcast. This may mean the last advert being truncated.

- Otherwise (i.e. `originalMediaObject` is an HTML5 video element and `newMediaObject` is a video/broadcast object) proceed with the switch but no timing requirements apply to the switch.

NOTE 5: Under these circumstances, a poor user experience may result.

5) At the last possible moment before the switch preparation deadline, the algorithm for attempting to allocate suitable video and audio decoders for `newMediaObject` is invoked.

8.1.4.3 Executing the switch

The executing the switch algorithm shall be followed on the first occasion that one of these conditions occurs for each call to the `switchMediaPresentation` method that has not failed with the `promise` being rejected or resolved.

- If `timelineSelector` is `null`, when the terminal detects the end of the media item in `originalMediaObject`.
- If `timelineSelector` is not null, when the terminal first detects that the time on the timeline indicated by `timelineSelector` has passed the time indicated by `switchTime` using the algorithm defined in clause 4.2.1.
- If `originalMediaObject` is an HTML5 video element and if the terminal detects 'ended playback' (as defined in HTML5 [3]) of the media item in `originalMediaObject` without the timeline indicated by `timelineSelector` passing the time indicated by `switchTime`.

The executing the switch algorithm shall consist of the following steps.

1) If any of the preconditions checked at the time `switchMediaPresentation` was originally called are no longer valid, then reject `promise` with an `InvalidStateError` and return.

2) If `minimumSwitchPerformanceRequired` is a URN that identifies a performance profile supported by the terminal and it is known to the terminal that the requirements of that performance profile will not be met then resolve `promise` with “NoPerformanceProfileMet” .

NOTE 1: See clauses 9.3.1, 9.3.2 and 9.3.3 for reasons why the requirements of a supported performance profile may, temporarily, not be met. In some implementations, meeting the requirements of a supported performance profile will only be possible if `newMediaObject` has both a suitable video decoder and a suitable audio decoder already allocated at this step in the algorithm. Other implementations may be able to meet the requirements of a supported performance profile even when video and audio decoders have to be released from `originalMediaObject` and re-used for `newMediaObject`, perhaps only under some specific circumstances such as the advert using the same codecs as the broadcast and the same or lower profile and level.

3) If `newMediaObject` does not have a suitable video decoder allocated and a suitable audio decoder allocated then perform the following substeps in order:

- Set the CSS `visibility` of `originalMediaObject` to `hidden`.
- If `originalMediaObject` is a video/broadcast object then put `originalMediaObject` in the stopped state and wait asynchronously for the state change to be completed (excluding execution of `onPlayStateChange` handler and dispatch of `PlayStateChange` event) allowing the browser task that invoked this algorithm to continue.
- If `originalMediaObject` is an HTML5 video element then pause `originalMediaObject` and the terminal shall wait asynchronously for the state change to be completed (excluding execution of any handler for the pause event) allowing the browser task that invoked this algorithm to continue.
- Take the video and audio decoders away from `originalMediaObject` and allocated to `newMediaObject`.

NOTE 2: The immediately above transition may not be frame accurate as quantified by the accuracy limit defined in clauses 4.2.3 and 4.2.5 of the present document.

4) If `newMediaObject` is a video/broadcast object and it is in the stopped state then start a transition to the presenting state. If it is already in the presenting state then it remains in that state.

5) If `newMediaObject` is an HTML5 media element which is paused then start playback as if the JavaScript `play` method had been called.

6) Wait asynchronously for `newMediaObject` to complete the transition to the presenting state (for a video/broadcast object) or to start playing (for an HTML5 video element) allowing the browser task that invoked this algorithm to continue.

7) If `newMediaObject` is a video/broadcast object and the transition to the presenting state failed with a `ChannelChangeError` then resolve `promise` with "VideoBroadcastPresentingFailed".

8) If `newMediaObject` is an HTML5 media element and an `error` event is fired then resolve `promise` with "MediaElementError".

9) Set the CSS visibility of `newMediaObject` to `visible` and set the audio volume to either 100 (for a video/broadcast object) or 1.0 (for an HTML5 video element).

10) Set the CSS visibility of `originalMediaObject` to `hidden` and set the audio volume to zero for that object.

NOTE 3: If `newMediaObject` is an HTML5 media element without a poster set, it may already be visible but transparent black until the play operation succeeds.

NOTE 4: Applications should not expect that modifying the CSS visibility of `originalMediaObject` after a switch will result both `originalMediaObject` and `newMediaObject` being visible. Some implementations may be able to decode more than one stream at one time but only output one of them. On such implementations, attempting this may result in `originalMediaObject` changing state away from presenting.

NOTE 5: Applications should not expect that changes to the audio volume of `originalMediaObject` after a switch will result both `originalMediaObject` and `newMediaObject` being audible. Even if it worked, it would likely give a poor user experience.

11) If `originalMediaObject` is a video/broadcast object and it is still in the presenting state then keep it in the presenting state.

12) If `originalMediaObject` is an HTML5 video element and it is not paused then pause it.

13) Resolve `promise` with `undefined`.

8.1.4.4 Allocate suitable video and audio decoders for `newMediaObject`

The terminal shall use the following algorithm for attempting to allocate suitable video and audio decoders for `newMediaObject`:

1) If `newMediaObject` already has a suitable video and a suitable audio decoder allocated then stop.

2) If the terminal has suitable video and audio decoders that are available to HbbTV but are not allocated to

anything then they shall be allocated to `newMediaObject` then stop.

- The availability (or otherwise) of video decoders shall be determined consistently with the values of the `extraSDVideoDecodes` and `extraHDVideoDecodes` properties as appropriate for the video to be presented by `newMediaObject` and taking into account the video being presented by `originalMediaObject`.
- If the terminal has more than one suitable video or audio decoder available to HbbTV but not allocated, it is implementation specific which are allocated.

3) If suitable video and audio decoders are allocated to an HTML5 video element that is paused, then take them from that video element and allocate them to `newMediaObject` then stop.
   - If more than one HTML5 video element is paused and has suitable video and audio decoders allocated then it is implementation specific which video element has the video and audio decoder taken from it.
4) If suitable video and audio decoders are allocated to a video/broadcast object that 1) is in the presenting state, 2) has CSS visibility of `hidden` and 3) is neither `newMediaObject` nor `originalMediaObject` then put that video/broadcast object into the stopped state. Once the state transition has finished, take the video and audio decoder from the video/broadcast object and allocate them to `newMediaObject` then stop.
5) If suitable video and audio decoders are allocated to `originalMediaObject` then stop.
6) Resolve `promise with "NoSuitableMediaDecoderAvailable"`.

9 System integration

9.1 MSE

9.1.1 Resource management

Except as follows, the requirements of clause 9.6 of TS 102 796 [1] shall apply to HTML5 media elements whose source is MSE as well as those whose source is defined in that document.

Multiple HTML5 media elements

The following requirement in clause 9.6.2 of TS 102 796 [1] does not apply to HTML5 media elements whose source is MSE.

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 present document does not require MSE implementations to support more than two SourceBuffer's - one for video and one for audio. As a consequence, there is no requirement corresponding to those quoted above for terminals to support more than one HTML5 media element whose src attribute refers to a MediaSource where, in turn, those MediaSources have SourceBuffers associated with it. Hence the advert insertion approach described in Annex J of TS 102 796 [1] is not required to work with MSE.

NOTE: Application can instead play multiple ads back-to-back in a single video element by appending them consecutively to the SourceBuffer(s). Clause 10.1.4 describes how AV sync alignment can be maintained when playing multiple ads consecutively via MSE in a single video element.

Preloading not interfering with broadcast

When broadcast video and audio is playing, if an application that has not called the `play` method (or equivalent such as using the `autoplay` attribute) creates one or more `SourceBuffer` objects and then appends data to them, this shall not disturb the broadcast video and audio presentation in any way.

9.1.2 System, video and audio formats

Terminals shall support playing via MSE all video and audio codecs listed in `<video_profile>` elements in their XML capabilities (see clause 10.2.4 of TS 102 796 [1]) except that, as there is no requirement to support more than one audio `SourceBuffer`, there is no requirement to support the MRMP mode of NGA audio with MSE even if it is supported for DASH.

Terminals shall support the video/mp4 and audio/mp4 byte stream formats defined in the MSE byte stream format registry as referenced through clause 10 of MSE [4].

9.1.3 Media Synchronization

Clause 9.7.1.2 of TS 102 796 [1] shall apply to HTML5 media elements whose source is MSE as well as those whose source is a URL. Specifically;

- If the media element is potentially playing and the readyState transitions from HAVE_FUTURE_DATA or more to HAVE_CURRENT_DATA or less (i.e. the application does not append data to a `SourceBuffer` fast enough) then the terminal shall behave as defined in Clause 9.7.1.2 of TS 102 796 [1] for when the terminal "is stalled while trying to fetch media data".

- If the readyState transitions from HAVE_CURRENT_DATA or less to HAVE_FUTURE_DATA or more (i.e. the application appends sufficient data to the `SourceBuffer`) then the terminal shall behave as defined in Clause 9.7.1.2 of TS 102 796 [1] for when it is no longer stalled because sufficient data has become available.

- The terminal shall be able to adjust presentation timing of the media element in order to maintain synchronisation (in the manner described in Clause 9.7.1.2 of TS 102 796 [1]) while sufficient data is available in the `SourceBuffer`.

- The application shall be able to determine what data it needs to append to the `SourceBuffer`, and when it needs to do so, by reading the `buffered` and `currentTime` properties of the media element and listening for when the seeking and progress events are fired by the media element. This shall include situations when the terminal adjusts the presentation timing of the media element in order to maintain synchronisation.

The requirements defined in this clause apply to HTML5 media elements used:

- as master media for inter-device synchronisation;

- for inter-device synchronisation when the terminal is acting as a slave terminal;

- as master media for multi-stream synchronisation; and

- as other media for multi-stream synchronisation.

Broadband streams that are delivered in response to `XMLHttpRequest` calls by an application and passed to the terminal using MSE shall be added to the list in clause 10.2.8.3 introduced by "Terminals shall support multi-stream synchronization with the constraints defined in clause 10.2.8.4 for broadband streams which are:"

NOTE: See clause 10.1.2 for the supported combinations of multi-stream sync using MSE.

9.1.4 Visibility and audibility

When content that has been delivered through MSE is played from the beginning to the end, the first and last frame of video shall be visible and the audio corresponding to the period from the start of the first video frame to the end of the last video frame shall be audible.

9.1.5 Out of band subtitles

Clause A.2.12.2 ("Out-of-band text tracks") of TS 102 796 [1] shall be supported for HTML5 video elements whose source is MSE.

9.2 Reliability and resilience

Terminals shall remain fully functional in the following circumstances. Fully functional in this case means at least that video, audio and subtitles are presented, the HbbTV application runs as specified including advert substitution and stream event reception:

- An application substitutes broadcast video and audio 20 times, returning to broadcast each time, where sometimes the substituted video and audio is delivered by MSE, sometimes by DASH and sometimes by non-adaptive HTTP streaming.
- The user changes the selected service 20 times between two services with different codecs (e.g. MPEG-2 SD and AVC HD for video) and both carrying broadcast-related autostart applications that substitute broadcast video and audio. The time interval between requested service changes is such that some service changes happen;
  - while a substitution has been scheduled but has not yet started
  - while a switch from broadcast to broadband is in progress
  - while substituted video and audio is being presented
  - while a switch from broadband to broadcast is in progress
  - after a substitution has completed and the broadcast video and audio is being presented

9.3 Limitations and restrictions

9.3.1 User-initiated background activities

Simultaneously with presenting broadcast television, terminals may enable the user to initiate background activities that, in turn, may reduce switching accuracy and/or increase switching duration while they are happening. Some possible examples of this include the following;

- Local PVR recording or time-shift
- Relaying broadcast content over broadband (for example using a technology such as SAT>IP).
- Picture in picture (e.g. displaying two broadcast services at the same time, one scaled to a small proportion of the display)
- The user opening the main UI of the TV at the exact time the switch is supposed to happen

Other examples of user-initiated background operations may exist.

It is implementation specific whether any particular background activity supported by a terminal interferes with meeting any switching performance requirement that the terminal implements. If the `minimumSwitchPerformanceRequired` argument is set to a profile where the terminal implementer can predict that the performance requirements will likely not be met due to a user-initiated background operation then the switch concerned shall be rejected.

9.3.2 Application-initiated activities

Activities initiated by an HbbTV application may reduce switching accuracy and/or increase switching duration while they are happening. Some possible examples of this include the following;

- Multi-stream synchronisation (as defined in clause 13.3.2 of TS 102 796 [])
- File download (as defined in annex H of TS 102 796 [])
- Particularly complex subtitle rendering shortly before or at the switch time

It is implementation specific whether any particular application-initiated activity interferes with meeting any switching performance requirement that the terminal implements. If the `minimumSwitchPerformanceRequired` argument is set to a profile where the terminal implementer can predict that the performance requirements will likely not be met due to an application-initiated background operation then the switch concerned shall be rejected.

9.3.3 System-initiated background activities

Simultaneously with presenting broadcast television, terminals may perform background activities that are not initiated by the user that may reduce switching accuracy and/or increase duration while they are happening. Some possible examples of this include the following;

- Downloading a software update prior to installation
- Acquiring EPG data (e.g. DVB-SI EIT schedule)

Other examples of system-initiated background activities may exist.

NOTE: Some terminals will defer some system-initiated background activities until after the terminal is put in active standby in order to give a better user experience. However system-initiated activities while presenting broadcast television are certainly possible.

It is implementation specific whether any particular background operation supported by a terminal interferes with meeting any switching performance requirement that the terminal implements. If the `minimumSwitchPerformanceRequired` argument is set to a profile where the terminal implementer can predict that the performance requirement will likely not be met due to a system-initiated background operation then the switch concerned shall be rejected.

9.3.4 Non-predictable background activities

Television receivers are complex systems with many activities and processes happening in parallel. Some will not be interruptible. The design of the Fast Media Switch API is intended to minimise the impact of this but eliminating it is impossible. There will always be some risk of non-predictable activities delaying one particular switch operation.

If a terminal detects that a switch operation from broadcast to broadband with the `minimumSwitchPerformanceRequired` argument set to the URN of a performance profile is being delayed and would be outside of the performance requirements for the indicated profile then it should abort the switch and not just go ahead anyway.

There is also a theoretical risk of not meeting a switching performance requirement because of an interruption by an unrelated activity or process in the middle of a switch, once `originalMediaObject` has been stopped or hidden but before `newMediaObject` has been started or shown.

9.3.5 Testing considerations

Testing of any performance requirements for switching accuracy and duration shall be done in the absence of both user-initiated background activities (as defined in clause 9.3.1 of the present document) and application-initiated activities (as defined in clause 9.3.2 of the present document) and, as far as can be known, in the absence of system initiated background activities (as defined in clause 9.3.3 of the present document).

Testing should take account of non-predictable background activities (as defined in clause 9.3.4 of the present document) and not require a 100% pass rate.

10 Capabilities

10.1 MSE

10.1.1 Basic requirements

Terminals shall support pause and resume of content delivered through MSE.

Terminals shall support random access into content delivered through MSE.

NOTE: The present document inherits the requirement from clause 1.2 of MSE [4] that "Implementations MUST support at least 1 MediaSource object with the following configurations:
- A single SourceBuffer with 1 audio track and/or 1 video track.
- Two SourceBuffers with one handling a single audio track and the other handling a single video track."
There is no requirement, either in the present document or inherited, to support SourceBuffers with text tracks.

10.1.2 Multi-stream media synchronization

The requirements of clause 10.2.8 of TS 102 796 [1] that relate to broadband streams shall apply to streams delivered by MSE.

In Table 14 of TS 102 796, "Mandatory combinations of media type, systems layer, timeline and delivery protocol", combinations 1 and 4 shall also be supported for audio delivered by MSE using the timeline for HTML media elements defined in clause 13.1.2. Table 1 below shows this as corresponding additions to Table 14 of TS 102 796. The contents of the new "MSE" column is blank for existing rows in Table 14.

Table 1: Additional rows and an additional "MSE" column for Table 14 "Mandatory combinations of media type, systems layer, timeline and delivery protocol" in ETSI TS 102 796 [1]

Delivery	**Broadcast**		**Progressive Streaming**			**MPEG DASH**	**MSE**	**HTTP Out of Band**	**Status**
Systems Layer	**MPEG2-TS**		**ISOBMFF**	**MPEG2-TS**		**ISOBMFF**	**ISOBMFF**		
Timeline for m/s sync	**PTS**	**TEMI**	**CTS**	**PTS**	**TEMI**	**DASH-PR**	**HTML-media**	**EBU-TT-D**	
N		Video					Audio		M
N+1		Video, Subtitles					Audio		M

10.1.3 Memory for MSE

Terminals shall have sufficient memory to load and run a broadcast-related application that does the following;

- Creates a video/broadcast object and connects it to a running HD broadcast using `bindToCurrentChannel`.
- While broadcast video is playing and without the broadcast video or audio or subtitles being disturbed;
  - Creates an HTML5 video element and a `MediaSource` and sets the source of the video element to the MediaSource.
  - Adds the video element to the DOM with the same parent as the video/broadcast object and immediately behind it on the CSS z-axis.
  - Creates a `SourceBuffer` for video and a `SourceBuffer` for audio.
  - Incrementally loads all of an advert into memory, in the form of an ISOBMFF file, that has the parameters specified in clause 4 of HbbTV Targeted Advertising part 2 [9].
    - Reading video and audio data using `XMLHttpRequest`
    - As data is read, appends video and audio to the respective `SourceBuffer` until either 1) all the data of each media type has been read and appended or 2) no more data can be appended
    - Releasing data in memory that has been successfully appended to a `SourceBuffer` such that it can be garbage collected.

NOTE 1: As explained in annex D reading an entire advert into RAM in a single `XMLHttpRequest` call is very inefficient and may result in a peak memory requirement that is twice the size of the advert. Incrementally reading an advert in smaller chunks reduces the peak memory requirement.

  - Calls `load` on the HTML5 video element
  - Waits for the HTML5 video element to reach HAVE_ENOUGH_DATA
- In parallel with the remaining steps, as space permits, appends any remaining video / audio to the respective source buffers until all of the advert has been appended and then calls `endOfStream` with the error parameter unset.
- Calls `switchMediaPresentation` with the video/broadcast object as `originalMediaObject`, the HTML5 video

element as `newMediaObject`,`timelineSelector` specifying a timeline in the broadcast, `timelineSource` as `true` and `switchTime` as a value more than 5 seconds ahead of the current timeline position of the broadcast.

  - Waits for the `promise` returned to be resolved with `undefined`.

- Calls `switchMediaPresentation` with the video/broadcast object as `newMediaObject`, the HTML5 video element as `originalMediaObject` `timelineSelector` as `null`, `timelineSource` as `true` and `switchTime` as zero.
- Waits for the `promise` returned from the second call to `switchMediaPresentation` to be resolved with `undefined`.

10.1.4 AV Sync

Multiple adverts can be played consecutively using a single video element by appending their data consecutively to the `SourceBuffer`(s) used by that video element. As noted in 4.2.5, the audio and video components of broadband media may not end at precisely the same moment. This could result in AV Sync drifting out of alignment.

The coded frame processing and audio splicing algorithms in MSE require implementations to adjust the presentation and decode timestamps of audio and video data when appended to a `SourceBuffer` by the amount specified by the application in the `timestampOffset` attribute. Applications can therefore ensure AV Sync remains in alignment by setting the `timestampOffset` attribute of the `SourceBuffer` before beginning to append data for the next advert in the sequence.

10.2 Media switching

10.2.1 Advert delivery

Terminals shall support switching from broadcast to any of the following and back to the broadcast. Terminals shall also support switching between any combinations of the following. For each of the following, terminals shall support switching at a defined time (`timelineSelector` from 8.1.3 is not null) and switching at the end of the media item (`timelineSelector` is null). Terminals shall support the new content being played from the beginning, from a static position and, for a dynamic DASH MPD, the live edge. Support for switching from one broadcast to another broadcast is not required.

- a media file for non-adaptive HTTP streaming
- a static DASH MPD
- a dynamic DASH MPD
- content being provided using MSE
- if the terminal supports the download option ("+DL" as defined in TS 102 796 [1]), a Download object

10.2.2 Broadcast formats

For each of the following combinations of broadcast video and audio codec that the terminal supports, it shall be able to switch to broadband-delivered content (MSE, DASH, non-adaptive HTTP streaming) encoded using AVC and HE-AAC (with and without subtitles) and return to the broadcast.

- MPEG-2 SD video + MPEG-1 layer 2 audio + broadcast subtitles
- MPEG-2 SD video + AC-3 audio + broadcast subtitles
- AVC SD/HD video + HE-AAC audio + broadcast subtitles
- AVC SD/HD video + E-AC-3 audio + broadcast subtitles
- AVC SD/HD video + AC-3 audio + broadcast subtitles
- HEVC HD SDR video + E-AC-3 + broadcast subtitles
- HEVC HD SDR video + HE-AAC + broadcast subtitles

- HEVC HD SDR video + AC-4 audio + broadcast subtitles
- HEVC HD SDR video + MPEG-H audio + broadcast subtitles
- HEVC HD HDR video (HLG10, PQ10) + E-AC-3 + broadcast subtitles
- HEVC HD HDR video (HLG10, PQ10) + HE-AAC + broadcast subtitles

For each of the following combinations of broadcast video and audio codec that the terminal supports, it shall be able to switch to broadband-delivered content (MSE, DASH, non-adaptive HTTP streaming) encoded using the same video and audio codec and return to the broadcast.

- HEVC HD SDR video + E-AC-3 + broadcast subtitles
- HEVC HD SDR video + HE-AAC + broadcast subtitles
- HEVC HD SDR video + AC-4 audio + broadcast subtitles
- HEVC HD SDR video + MPEG-H audio + broadcast subtitles

10.2.3 Timelines

Timelines shall be supported as follows.

- When `originalMediaObject` is a video/broadcast object, PTS ("urn:dvb:css:timeline:pts") and TEMI ("urn:dvb:css:timeline:temi:<component_tag>:<timeline_id>") shall be supported.. For more details, see TS 103 286-2 [i.5] as referenced from TS 102 796 [1].
- On terminals that are able to maintain state relating to broadcast video and audio after the end of a switch away from broadcast, when `newMediaObject` is a video/broadcast object, PTS ("urn:dvb:css:timeline:pts") and TEMI ("urn:dvb:css:timeline:temi:<component_tag>:<timeline_id>") shall be supported. Support for these timelines on terminals that cannot maintain such state is probably not possible.
- When an ISOBMFF file delivered by non-adaptive HTTP streaming is presented by an HTML5 video element, ISOBMFF composition time ("urn:dvb:css:timeline:ct") shall be supported.
- When streaming content delivered by DASH is presented by an HTML5 video element, period relative Timeline ("urn:dvb:css:timeline:mpd:period:rel:<ticks-per-second>" or "urn:dvb:css:timeline:mpd:period:rel:<ticks-per-second>:<period-id>") shall be supported. For more details, see TS 103 286-2 [i.5] as referenced from TS 102 796 [1].
- When content delivered via broadband and MSE is presented by an HTML5 video element, the media timeline of the media resource of an HTML media element shall be supported as defined in clause 4.8.12.6 of the HTML specification [3] and clause 13.1.2 of the present document.

10.2.4 Switch preparation deadline

Terminals may require that the `switchMediaPresentation` method be called some time in advance of the time at which the switch is to happen – the switch preparation deadline. If terminals require this then they shall not require more than 2 seconds advance notice.

NOTE 1: It is recommended that broadcasters should call the `switchMediaPresentation` method at least 5 seconds in advance of the time at which the switch is to happen unless a shorter notice period is necessary – e.g. for advert replacement in live content. Implementations may use algorithms which give shorter or more accurate transitions when more advance notice is given.

When `timelineSelector` refers to a TEMI timeline, the 2 seconds advance notice is in addition to up to 2.5 seconds that may be needed to start monitoring the TEMI timeline as defined in clause xx.yy of TS 102 706 [].

NOTE 2: The TEMI timeline may already be monitored, e.g. if the application has called `MediaSynchroniser.initMediaSynchroniser` for that timeline.

10.2.5 Audio considerations

Changes of the audio output format (codec, channel configuration, sampling rate, altered frame timing) may create discontinuities on a digital audio output interface. In order to mitigate audio artefacts from discontinuities on a receiver connected through a digital output, it is strongly recommended that the output format on the digital output interface is derived from the capabilities of the sink device and kept constant also during the advert.

10.3 Minimum terminal capabilities

10.3.1 General

Implementations of the present document shall indicate this by the addition of one or more `<ta>` elements in the XML device capabilities as defined in clause A.2.1.

The `version` attribute shall indicate "1.1.1" for implementations of the present document.

Zero, one or more `profile` elements shall be present as children of the `ta` element to identify profiles that the terminal meets the requirements of.

If the terminal meets the requirements for profile 1 as defined in clause 5.3.2 of HbbTV Targeted Advertising part 2 then a `profile` element containing the URN identifying that profile shall be included. Otherwise such a `profile` element shall not be included.

If the terminal meets the requirements for profile 2 as defined in clause 5.3.3 of HbbTV Targeted Advertising part 2 then at least two `profile` elements shall be included, one containing the URN for profile 1 and one containing the URN for profile 2. If the terminal does not meet the requirements for profile 2 then a `profile` element containing the URN for profile 2 shall not be included.

Additional `profile` elements may be included containing URNs identifying other profiles. URNs for profiles defined in TS ? ?-2 shall be listed first.

The `broadcastTimelineMonitoring` attribute shall be `true` if monitoring a broadcast timeline (TEMI or PTS) is supported while broadband content is being presented (see clause 10.2.3) and shall be `false` if the terminal does not support this monitoring.

The `GOPIndependentSwitchToBroadcast` attribute shall be `true` if the terminal is able to maintain state relating to broadcast video and audio after the end of a switch from broadcast to broadband (see clause 10.4) such that the duration of switches back to the broadcast are independent of the GOP length (see clause 4.2.6). It shall be `false` if the terminal is not able to maintain that state.

NOTE: To monitor a broadcast timeline does not necessarily require a second audio and video decoder if the terminal can present a broadband stream while simultaneously demultiplexing and extracting PCR, PTS and TEMI data from the broadcast MPEG-2 transport stream and performing the processes the terminal uses to time presentation of broadcast access units (such as regenerate the STC) - see Clause D.1.2 of ISO 13818-1 [ref]. If the broadband stream is also an MPEG-2 transport stream, then the terminal would need to be capable of demultiplexing two MPEG transport streams simultaneously and running the processes for timing presentation for both.

10.4 Single and multiple video and audio decoders

The present document can be implemented on terminals with only a single video decoder and a single audio decoder as well as terminals with more than one of each. Annex F provides some background on scenarios for how these can be used.

The following shall apply to terminals with only a single video decoder and a single audio decoder.

- Clause 8.1.4.3 defines how the single video and audio decoders shall be re-used when switching from `originalMediaObject` to `newMediaObject`.
- When switching from broadcast to broadband, re-using the single video decoder and the single audio decoder will inevitably mean loss of state relating to broadcast video and audio after the end of the switch. The performance requirements defined in ??? ??? - 2 [9] shall apply when switching to broadband content at the start of the broadband content and at positions where seeks are required to be performed precisely according to

clause 9.1.3 of TS 102 796 [1] and do not apply at other positions in the broadband content.

NOTE 1: This will mean the following switch from broadband back to broadcast will have to wait for a GOP or other entry point (see clause 4.2.6).

- Concerning clause 10.2.3, a consequence of the loss of the ability to monitor the broadcast video and audio is the terminal being unable to monitor a PTS or TEMI timeline while playing an advert.
- Profile 1 defined in clause 5.3.2 of HbbTV Targeted Advertising part 2 [9] shall be supported.

The following shall apply to terminals with more than one video decoder and more than one audio decoder.

- Clause 8.1.4.4defines how the multiple video and audio decoders shall be used when switching from `originalMediaObject` to `newMediaObject`.
- Step 11 in clause 8.1.4.3 defines that, when the multiple video and audio decoders are be used for switching from broadcast to broadband, the broadcast shall continue to be decoded after the switch has completed.
- The performance requirements in ??? ??? - 2 [9] shall apply regardless of the position in the broadband content only if decoder resources are already allocated to it and it is ready to resume playback from that position.

NOTE 2: An application can ensure playback is ready to resume from a specific time position by commencing playback before that time position and once playback has started, putting the video element into the paused state at the required time position.

- Terminals shall be able maintain state relating to broadcast video and audio after the end of the switch. Hence the following switch from broadband back to broadcast shall not have to wait for a GOP or other entry point (see clause 4.2.6).
- Concerning clause 10.2.3, terminals shall be able to monitor PTS and TEMI timelines in the broadcast.
- Profile 2 defined in clause 5.3.2 of HbbTV Targeted Advertising part 2 [9] shall be supported.

Terminals with more than one video decoder and more than one audio decoder may be unable to use more than one of each for targeted advertising under certain circumstances including;

- When the terminal is decoding UHD video (see clause 5.2.2 of HbbTV Targeted Advertising part 2 [9]).

NOTE 3: Clause A.2.2 uses the example of a terminal able to simultaneously decode either one UHD stream or 2 HD streams or 1 HD stream and 2 SD streams or 4 SD streams (with audio in all cases).

- When the terminal is performing some user-initiated background activities such as PVR or picture in picture (see clause 9.3.1)
- When the terminal is performing some application-initiated background activities such as multi-stream synchronisation (see clause 9.3.3).

NOTE 4: On a profile 2 terminal with more than one video decoder and more than one audio decoder, if only one of each is available, the switching performance may be worse than profile 1. Under these circumstances, a switch with `minimumSwitchPerformanceRequired` set to the URN for either profile 1 or profile 2 would fail.

The requirements in clause 6.2.2.7 of TS 102 796 [1] concerning demultiplexing and processing of data carried in MPEG-2 sections in the broadcast shall apply regardless of the number of video and audio decoders in a terminal.

11 Security

The present document defines extensions to TS 102 796 [1] that provide additional facilities that can be used by applications. The fundamental aspects of security are inherited from that specification - application signalling, application transport and the use of TLS for security over broadband connections.

The extensions defined in the present document can also be used by operator applications as defined in TS 103 606 [i.3]. In this case, the fundamental aspects of security are inherited from that specification - application discovery, authentication and installation.

One feature of TS 102 796 [1] likely to be widely used in the context of targeted advertising is DSM-CC stream events. TS 102 796 [1] includes the following warning about these which remains valid.

> Application developers should be aware that in some circumstances an attacker may be able to modify the broadcast signalling from which this data is derived. Applications shall not use this data in a way that would result in it being executed by the browser. Applications should be written to be tolerant of incorrectly formatted data or values for this data which are outside the expected range without hanging up or crashing.

The present document improves security relative to TS 102 796 [1] by including the Web Cryptography API (see clause A.3.3). If the system generating DSM-CC stream events would add authentication inside the stream event payload then an application could use this API to verify part of the payload. If the system generating DSM-CC stream events would encrypt part of the stream event payload then an application could use this API to decrypt the encrypted payload. The present document does not require support for encryption and/or authentication of the elementary streams carrying stream events.

In addition, all traffic to and from applications over the broadband interface should be secured.

- Applications downloaded over broadband need to be downloaded using TLS.
- Requests to advert decision servers need to be done over TLS and responses returned via TLS.
- Mixed content requirements [i.8] will mean that adverts to be played via MSE will need to be downloaded via TLS.
- Careful consideration should be given to the use of TLS for adverts played using DASH.
- Reporting needs to be done via TLS.

12 Privacy

12.1 General

Privacy in the context of targeted advertising in general is an important subject and in the context of broadcast television possibly even more so, given that a broadcast-related autostart DAS app may start without the viewer being aware (e.g. in the absence of a red button).

The present document has been developed with the view that the main privacy and data protection responsibilities are assumed by the broadcaster and/or operator who is providing the DAS app, therefore it is they who should ensure that the app is engineered to comply with all applicable legislation and regulation (e.g. the GDPR).

While limited targeting of advertising may be possible without communication of personal data, many use cases will require identifying a specific person or household. Under such circumstances broadcasters should consider whether consent is required to opt-in or out for this level of targeting.

When advert substitution is done well, viewers need not be aware that their broadband connection is being used to deliver an advert instead of the advert coming via the normal broadcast route. In some markets viewers may not reasonably expect their broadband bandwidth to be consumed (e.g. where the viewer has a broadband data cap) and in general there may be congestion on the broadband connection with the associated impact to the viewing experience (e.g. where the viewer has a limited broadband bandwidth connection). For these reasons broadcasters should consider providing a means of removing consent, irrespective of their position according to the appropriate regulation. Broadcasters should also recognise that if consent is on an opt-out basis there would be a responsibility to make it clear to the viewer how this can be achieved.

Care should be taken in the DAS app to store and transmit in a secure manner anything used in the process of targeting and measurement which may be considered personal data.

12.2 Cookies

In the context of the DAS app, a broadcaster's responsibility includes ensuring data held locally in persistent cookies or local storage can be removed should the viewer wish. Note that, while persistent cookies cannot directly be removed by an app, the expiry date may be modified to be in the past thus indicating to the terminal that they are to be removed (see step 11 of clause 5.3 of RFC6265 [i.13].

Terminals shall support third party cookies and shall not block all third party cookies by default.

NOTE 1: It is hoped to define a replacement for the above in a future version of the present document that is compatible with regulation and current industry practice but still permits frequency capping of adverts across multiple broadcasters and optimising processes for obtaining and recording consent.

NOTE 2: Clause 12.1.2 of TS 102 796 [1] permits but does not require blocking of all third party cookies. Such blocking is excluded by the present document. The additional requirement from that clause remains applicable – for terminals provide the user with the option of blocking third party cookies.

13 Media Synchronization

13.1 MSE

13.1.1 Use of media elements whose source is MSE

In all places where an application is permitted to use an HTML5 media element in Media Synchronization, it shall be able to use an HTML5 media element whose source is an MSE `SourceBuffer` as an alternative to an HTML5 media element whose source is a URL.

13.1.2 Timeline Selector for media timeline of media elements

The Timeline Selector `urn:hbbtv:sync:timeline:html-media-timeline` shall refer to the media timeline of the media resource of an HTML media element as defined in clause 4.8.12.6 of the HTML specification [3].

NOTE 1: The timeline specified by this Timeline Selector can be used with an HTML media element in the situation where the media resource is being provided by the application via Media Source Extensions. Other Timeline Selectors that depend on the container format of the media resource cannot be used in this situation.

The use of this Timeline Selector shall be supported for when an HTML media element is used:

- with the APIs defined in the present document and
- for media synchronisation as described in clause 13 of TS 102 796 [1] including:
  - as the timeline for the master media for inter-device synchronisation and
  - as the timeline for the master media for multi-stream synchronisation and
  - as the timeline for the other media for multi-stream synchronisation
  - for application to media synchronisation.

NOTE 2: Support for media synchronisation means that this Timeline Selector can be used with the `initMediaSynchroniser()` and `addMediaObject()` methods of the Media Synchroniser API (as defined in clause 8.2.3 of TS 102 796 [1]) and also via the Timeline synchronisation service endpoints (CSS-CII and CSS-TS as defined in clause 13.8 of TS 102 796 [1]).

Table 18 in clause 13.4.2 of TS 102 796 [1] shall be extended with an extra row as shown in table 2 below:

Table 2: Additional row for Table 19 of the HbbTV 2 specification

Types of timeline	Supported for
Media timeline of the media resource of an HTML media element	Any media played in an HTML media element

Values on the timeline represented by this Timeline Selector shall have a tick rate of 1000 Hz and therefore be expressed in units of milliseconds when used:

- in `CorrelationTimestamp` objects (defined in the clause 8.2.3.3 of TS 102 796 [1]); and
- in messages exchanged via the Timeline Synchronisation service endpoint (defined in clause 13.8 of TS 102 796 [1]).

14 Companion screens

The present document does not introduce any additional requirements beyond those in TS 102 796 [1].

Annex A (normative): Web and OIPF specification profile

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

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

Table A.1: Section-by-section profile of the OIPF DAE specification

Section, sub-section	Reference in DAE [2]	Status in HbbTV®	Status in the present document
Object Factory API	7.1	M(*)	An additional method shall be supported as follows; `MediaSwitcher createMediaSwitcher()` Description: Creates a new `MediaSwitcher` embedded object.
The LocalSystem class	7.3.3	NI	Read-only access to the following property shall be supported: • `mute` Note that operator applications as defined in TS 103 606 [i.3] are required to have read-write access to this property under circumstances defined in that document. That requirement shall only apply to operator applications and not for regular HbbTV applications.

A.2 Modifications, extensions and clarifications to volume 5

A.2.1 Extensions to the OIPF-defined capability negotiation mechanism

The following schema is an extension of the schema defined by clause A.2.15 of TS 102 796 [1] which in turn extends annex F of the OIPF DAE specification [2]. The extensions introduced in the present document are underlined – this is purely for ease of understanding and has no technical consequences. The normative definition of this schema is found in the electronic attachments - see Annex B of the present document.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:hbbtv="urn:hbbtv:config:oitf:oitfCapabilities:2017-1"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="urn:hbbtv:config:oitf:oitfCapabilities:2017-1"
           xmlns:oipf="urn:oipf:config:oitf:oitfCapabilities:2011-1"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified" version="2019.1" vc:minVersion="1.1"
           xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning">
    <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:attribute name="hdr" type="xs:anyURI" 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:element name="video_display_format" type="hbbtv:videoDisplayFormatType"
                                minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="broadcast" type="xs:anyURI" minOccurs="0"
                                maxOccurs="unbounded"/>
                    <xs:element name="applicationDiscovery" type="xs:anyURI" minOccurs="0"
                                maxOccurs="unbounded"/>
                    <xs:element name="hdmi" type="hbbtv:hdmiType"/>
                    <xs:element name="ta" type="hbbtv:taType" minOccurs="0"/>
                    <xs:any namespace="##any" processContents="lax"/>
                </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:simpleType name="colorimetryListType">
        <xs:list itemType="hbbtv:colorimetryType"/>
    </xs:simpleType>
    <xs:simpleType name="colorimetryType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="bt709"/>
            <xs:enumeration value="bt2020"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="videoDisplayFormatType">
        <xs:attribute name="width" type="xs:integer" use="required"/>
        <xs:attribute name="height" type="xs:integer" use="required"/>
        <xs:attribute name="frame_rate" type="xs:integer" use="required"/>
        <xs:attribute name="bit_depth" type="xs:integer" use="required"/>
        <xs:attribute name="colorimetry" type="hbbtv:colorimetryListType" use="required"/>
    </xs:complexType>
    <xs:complexType name="hdmiType">
        <xs:attribute name="broadbandOverlay" type="xs:boolean" use="required"/>
        <xs:attribute name="monitoringWhileBroadband" type="xs:boolean" use="required"/>
        <xs:attribute name="scaling" type="xs:boolean" use="required"/>
    </xs:complexType>
    <xs:complexType name="taType">
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element name="profile" type="xs:anyURI"/>
        </xs:sequence>
        <xs:attribute name="version" type="hbbtv:versionType" use="required"/>
        <xs:attribute name="broadcastTimelineMonitoring" type="xs:boolean" use="required"/>
        <xs:attribute name="GOPIndependentSwitchToBroadcast" type="xs:boolean" use="required"/>
    </xs:complexType>
    <xs:simpleType name="versionType">
        <xs:restriction base="xs:string">
            <xs:pattern value="[1-9][0-9]*\.[1-9][0-9]*\.[1-9][0-9]*"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

Here is an example of XML capabilities including these extensions.

```
<profilelist xmlns="urn:hbbtv:config:oitf:oitfCapabilities:2017-1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:hbbtv:config:oitf:oitfCapabilities:2017-1 hbbtv-capabilities-2019.xsd">
    <ui_profile name="OITF_HD_UIPROF+DVB_S+TRICKMODE">
        <ext>
            <parentalcontrol schemes="dvb-si">true</parentalcontrol>
            <clientMetadata type="dvb-si">true</clientMetadata>
            <temporalClipping/>
            <html5_media>true</html5_media>
            <ta version="1.1.1" broadcastTimelineMonitoring="true"
                GOPIndependentSwitchToBroadcast="true">
                <profile>urn:hbbtv:ta:profile:2019:2</profile>
            </ta>
        </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"/>
</profilelist>
```

A.2.2 Extensions and clarifications to the application/oipfCapabilities embedded object

The properties `extraSDVideoDecodes`, `extraHDVideoDecodes` shall be modified as follows;

- The values returned shall reflect the number of additional streams containing video accompanied by audio that are possible to decode and present. If decoding a video stream is possible but not accompanied by an audio stream then the decoding of that video stream shall not be included in the value returned. Video streams that can be decoded but not presented shall not be included in the value returned.
- If the value returned is non zero then a call to play an A/V Control object or `video/broadcast` object or an HTML5 video element shall not fail due to lack of availability of media decoding resources if the call is made in that same spin of the event loop and if the video to be played is SD (for `extraSDVideoDecodes`) or HD (for `extraHDVideoDecodes`).

The `application/oipfCapabilities` embedded object shall be extended with a property and two methods as follows.

`readonly Number` **`extraUHDVideoDecodes`**
This property holds the number of additional streams containing UHD video accompanied by audio that are possible to decode. Depending on the current usage of system resources this value may vary. The value of this property is likely to change if an SD or HD video is started. If decoding a video stream is possible but not accompanied by decoding an audio stream then the decoding of that video stream shall not be included in the value returned. Video streams that can be decoded but not presented shall not be included in the value returned. If the value returned is non zero then a call to play an A/V Control object or `video/broadcast` object or an HTML5 video element shall not fail due to lack of availability of media decoding resources if the call is made in that same spin of the event loop. Otherwise playing an A/V Control object or `video/broadcast` object or an HTML5 video element may still fail, even if `extraUHDVideoDecodes` was larger than 0 when last read. For A/V Control objects, in case of failure the play state for the A/V Control object shall be set to 6 (‘`error`’) with a detailed error code of 3 (‘`insufficient resources`’). For `video/broadcast` objects, in case of failure the play state of the `video/broadcast` object shall be set to 0 (‘`unrealized`’) with a detailed error code of 11 (‘`insufficient resources`’). For an HTML5 video element, the request to present media through the media element shall MediaError with code MEDIA_ERR_DECODE.

`StringCollection` **`broadbandCapabilities`**`( Number decoderIndex )`	
Description	Returns the maximum (static) broadband media decoding capabilities for the indicated decoder. These may not be available at a particular moment in time. A dynamic indication of what is available is provided by the properties `extraSDVideoDecodes`, `extraHDVideoDecodes` and `extraUHDVideoDecodes`. This method shall only return stream decoding capabilities and shall ignore a terminal’s capabilities to output streams simultaneously. Terminals may be able to decode more streams simultaneously than they can output. Media decoders shall be numbered from 1 increasing in steps of 1 up to the total number of media decoders in the terminal. Some terminals may not have discrete media decoders but a single multi-stream media decoder, in this case each stream that can be decoded should be represented as a separate media decoder. The numbering shall be consistent between this method and the `broadcastCapabilities` method. If `decoderIndex` is greater than the number of media decoders then null shall be returned. Decoders shall be ordered in decreasing capabilities, i.e. decoders supporting UHD shall be listed before those not

	supporting UHD and decoders supporting HD shall be listed before those not supporting HD. Each String returned shall be one of the <video_profile> elements returned by the `xmlCapabilities` property. Applications have no access to individual decoders. They are expected to call this method for all possible media decoders and aggregate the results in order to determine if a terminal would ever be able to decode more than one stream of a particular type. For example, on a terminal where decoder 1 supports UHD but decoder 2 does not, a call to the `switchMediaPresentation` method to switch from UHD to UHD will never be accelerated by using multiple decoders.	
Arguments	*decoderIndex*	A number identifying one of the media decoders in the terminal.

`StringCollection` **`broadcastCapabilities`**`( Number decoderIndex )`		
Description	Returns the maximum (static) broadcast media decoding capabilities for the indicated decoder. These may not be available at a particular moment in time. A dynamic indication of what is available is provided by the properties `extraSDVideoDecodes`, `extraHDVideoDecodes` and `extraUHDVideoDecodes`. This method shall only return stream decoding capabilities and shall ignore a terminal's capabilities to output streams simultaneously. Terminals may be able to decode more streams simultaneously than they can output. Media decoders shall be numbered from 1 increasing in steps of 1 up to the total number of media decoders in the terminal. Some terminals may not have discrete media decoders but a single multi-stream media decoder, in this case each stream that can be decoded should be represented as a separate media decoder. The numbering shall be consistent between this method and the `broadbandCapabilities` method. If `decoderIndex` is greater than the number of media decoders then null shall be returned. Decoders shall be ordered in decreasing capabilities, i.e. decoders supporting UHD shall be listed before those not supporting UHD and decoders supporting HD shall be listed before those not supporting HD. Each String returned shall be one of the URNs returned as the value of a <broadcast> element returned by the `xmlCapabilities` property. Applications have no access to individual decoders. They are expected to call this method for all possible media decoders and aggregate the results in order to determine if a terminal would ever be able to decode more than one stream of a particular type. For example, on a terminal where this method returns an empty `StringCollection` for decoder 2 then a call to the `switchMediaPresentation` method to switch from broadband to broadcast would not be accelerated by multiple decoders if decoder 1 was already used for the broadband.	
Arguments	*decoderIndex*	A number identifying one of the media decoders in the terminal.

Table A.2 summarises what would be returned by the above methods for an example terminal able to simultaneously decode either one UHD stream or 2 HD streams or 1 HD stream and 2 SD streams or 4 SD streams (with audio in all cases). Table A.3 shows an example of the values of the `extraUHDVideoDecodes, extraHDVideoDecodes` and `extraSDVideoDecodes` properties on such a terminal.

Table A.2: Summary of return values for `broadcastCapabilities` and `broadbandCapabilities` methods for 4 decoder example terminal

Method call with decoderIndex argument	Returned StringCollection
`broadcastCapabilities(1)`	Includes URN for UHD
`broadcastCapabilities(2)`	Includes URN for HD but not for UHD
`broadcastCapabilities(3)`	Includes URN for SD but not for for UHD or HD
`broadcastCapabilities(4)`	Includes URN for SD but not for for UHD or HD
`broadbandCapabilities(1)`	Includes <video_profile> element for UHD
`broadbandCapabilities(2)`	Includes <video_profile> element for HD but not for UHD
`broadbandCapabilities(3)`	Includes <video_profile> element for SD but not for UHD or HD
`broadbandCapabilities(4)`	Includes <video_profile> element for SD but not for UHD or HD

Table A.3: Examples of `extraUHDVideoDecodes`, `extraHDVideoDecodes` and `extraSDVideoDecodes` properties for 4 decoder example terminal

Content being decoded	UHD	HD	SD
`extraUHDVideoDecodes`	0	0	0
`extraHDVideoDecodes`	0	1	1
`extraSDVideoDecodes`	0	2	3

NOTE: Although the above example focuses on resolutions, other limitations may apply. For example, it may only possible to decode one HDR stream or to decrypt one stream at one time.

A.3 Modifications, extensions and clarifications to volume 5a

A.3.1 General

The following modifications shall be made to the profile of HTML5 [3] and other web standards defined by the OIPF Web Standards TV Profile [i.1].

A.3.2 Media Source Extensions

The W3C Media Source Extensions [4] shall be supported. The `changeType` method as defined in WICG [5] and further explained in [i.2] is optional. If the `changeType` method is present but does not work as defined then a `NotSupportedError` exception shall be thrown when the method is called.

NOTE: Some implementations may attempt to very aggressively evict data and never throw a `QuotaExceededError`. Applications should take care that data which is still to be played has not been evicted without the app noticing. Applications can minimise the risk of this by not appending data more than 30s in advance of the current play position and by checking the `HTMLMediaElement.buffered` property to confirm that the `SourceBuffer` contains what is expected once an `updateend` event has been received.

A.3.3 Web Cryptography API

The Web Cryptography API [6] shall be supported including at least all the algorithms that are suggested in clause 18.5.2 of that document. Support for 192-bit AES keys is optional.

A.3.4 VTTCue

The `VTTCue` class (see clause 9.1 of WebVTT [10]) shall be supported as follows;

- It shall be possible to construct instances of the `VTTCue` class using the constructor
- It shall be possible for HbbTV applications to get and set properties on such instances

- It shall be possible to add such instances to `TextTracks` using the `addCue` method as defined for `TextTrackCue` in HTML5 [3].
- It shall be possible to remove such instances from `TextTracks` using the `removeCue` method as defined for `TextTrackCue` in HTML5 [3].
- The `onenter` and `onexit` handlers shall be called for such instances according to the "time marches on" algorithm in HTML5 [3].

NOTE: The above is believed to be the minimum support for `VTTCue` necessary to enable JavaScript rendering of TTML subtitles using approaches like that taken by imscJS [i.14].

The present document neither requires nor excludes support for more of WebVTT [10] than listed above. HbbTV applications shall not rely on more of that specification than the above being implemented or indeed working correctly even if implemented.

A.3.5 navigator.cookieEnabled

The property "navigator.cookieEnabled" defined in the WHATWG HTML Living Standard [14] shall be supported and deliver reliable results. Specifically, if set to "true" then persistent storage of cookies shall be available and not only session cookies.

Terminals may implement this property from any snapshot of the WHATWG living standard [15] from the first commit in 2018 onwards.

Annex B (normative): Electronic attachments

The present document includes an electronic attachment ts_??????v010101p0.zip with the following contents.

- hbbtv-capabilities-2019-2.xsd
  This is the normative XML schema file whose text is included informatively in annex A.2.1 to the present document.
- example-ta.xml
  This is an example XML capabilities including the XML schema extensions defined in clauses 10.3.1 and A.2.1 of the present document and which validates using the schema.

The following other XML schema dependencies can be found in the electronic attachments for TS 102034 [11].

- sdns_v1.4r13.xsd
- sdns_v1.5r25b.xsd
- tva_metadata_3-1_v131.xsd
- tva_metadata_3-1_v171.xsd
- tva_mpeg7.xsd
- tva_mpeg7_2008.xsd
- xml.xsd

XML schema dependencies for the OIPF DAE specification [2] can be found at http://www.oipf.tv/docs/OIPF-Schemas_v2_3-2014-01-24.ZIP .

Annex C (informative) Timings and terminology

C.1 A model for broadcast timings based on switching performance

Signalling informs the DAS application of an Ad Placement Opportunity. This defines a window of time (in terms of the broadcast) during which the application can perform the substitution by showing broadband substitution content in place of broadcast content.

Figure C.1 and Figure C.2 illustrate the potential timing variation at the terminal when an application requests a switch from broadcast to broadband substitution content and back to broadcast again at times that the application specifies. The timing of the switch back from broadband to broadcast can be expressed in terms of the broadband content timeline as shown in Figure C.1; or it can be expressed in terms of the broadcast content timeline as shown in Figure C.2.

It is assumed that the timings conveyed by the signalling are accurate.

A terminal might not guarantee that the switches between broadcast and broadband-delivered content are instantaneous, are perfectly accurately timed, and that there is no drift in the rate of presentation of broadband content relative to the broadcast. This potential variation can be modelled by two scenarios representing the fastest and slowest manifestations of the process.

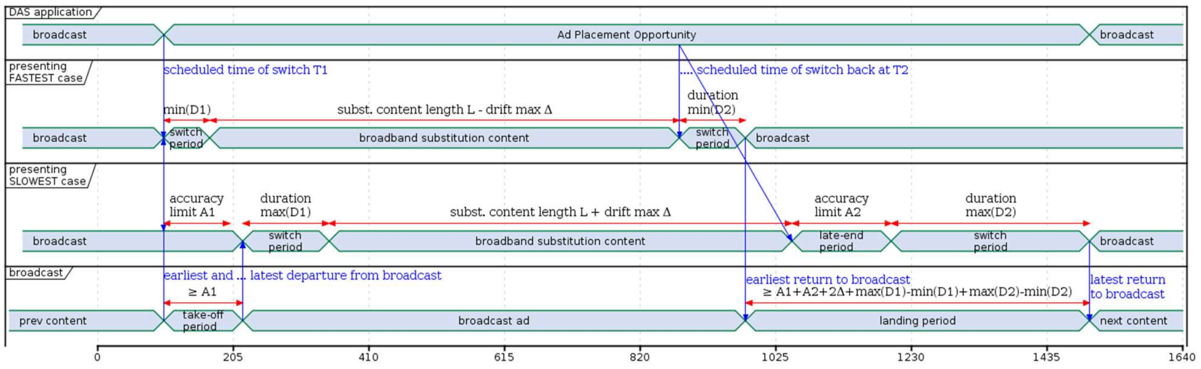


Figure C.1: Fastest and slowest timing scenarios for switching, when the switch back to broadcast is defined in terms of the broadband timeline.

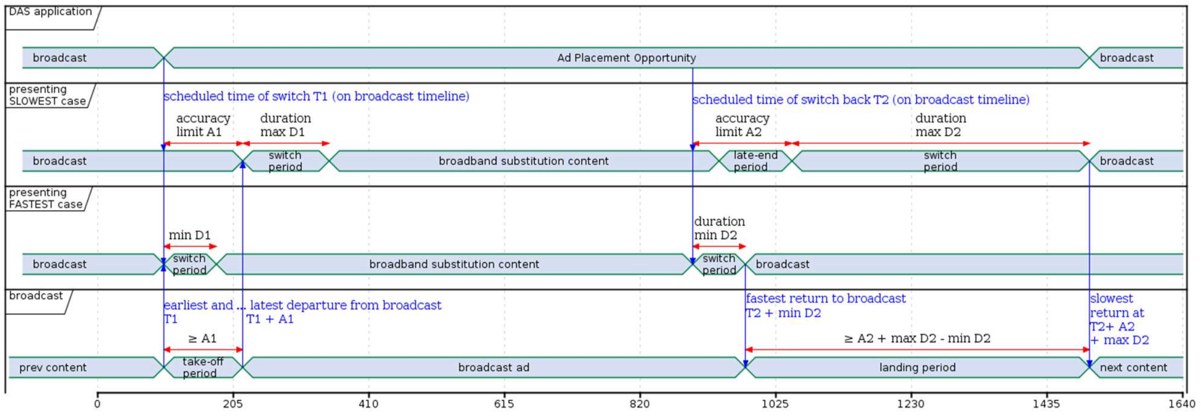


Figure C.2: Fastest and slowest timing scenarios for switching, when the switch back to broadcast is defined in terms of the broadcast timeline.

The application will specify to the terminal that the broadcast to broadband switch is to be scheduled to take place at a time *T*1. This is specified as a timestamp on the PTS or TEMI broadcast timeline. In the fastest scenario the switch will happen at exactly timestamp *T1*. In the slowest scenarios it will happen at *T*1 + *A*1 where *A*1 is an interval of time denoting the broadcast to broadband switching accuracy limit. This switch does not occur earlier than time *T*1.

When this switch commences, there could then be a broadcast to broadband switching period of black frames and silence before the substitution content begins to be presented, of duration *D*1. The fastest scenario corresponds to min *D*1 being the minimum duration of this period that the terminal can achieve. The slowest scenario corresponds to max *D*1 being the maximum duration of this period that the terminal can achieve.

The presentation of the substitution content might not be clock synchronised with the presentation of the broadcast content. This could result in the substitution content playing very slightly faster or slower than the broadcast. The amount of drift Δ is defined as the accumulated difference in elapsed time compared to if the presentation had been perfectly synchronised. In the fastest scenario, drift results in a shorter duration. In the slowest scenario, drift results in a longer duration.

The application will instruct the terminal to switch back to broadcast at a timestamp *T*2. This could be specified either in terms of the timeline of the substitution content (as shown in Figure C.1), or the timeline of the broadcast content (as shown in Figure C.2). Terminal capabilities will dictate which of these can be done.

L corresponds to the duration of broadband content that ideally would be played if there is perfect clock synchronisation and the switch occurs at exactly time *T*2. If the switch is specified in terms of the timeline of the substitution content then the corresponding elapsed time on the broadcast timeline is between *L* - Δ in the fastest scenario and *L* + Δ in the slowest scenario. For a terminal capable of a drift not exceeding δ ppm and given a broadcast with timing tolerance of ±30ppm (as specified by ISO 13818-1 [ref]) then:

$$\Delta = L\frac{30 + \delta}{1\,000\,000}$$

The broadband to broadcast switch could occur at exactly time *T*2 or might also be delayed by up to *A*2 milliseconds where *A*2 is broadband to broadcast switching accuracy limit. This could result in a late-end period of up to *A*2 milliseconds after the final frame of the substitution content was due to be presented but before the process of the switch back to broadcast has commenced. During this late end period, the terminal could, for example, present silence and a freeze frame of the final frame of the substitution content or black.

Once the switch back commences, there could be a broadband to broadcast switching period of black frames and silence before presentation of broadcast resumes. This duration of this period is *D*2. The fastest scenario corresponds to min *D*2 being the minimum duration of this period that the terminal can achieve. The slowest scenario corresponds to max *D*2 being the maximum duration of this period that the terminal can achieve.

NOTE: Switching accuracy and duration are defined separately for the switch from broadcast to broadband and the switch back to broadcast.

NOTE 2: The minimum and maximum achievable switching durations can be affected by the interval between random access points in the broadcast stream, depending on terminal capabilities. A longer interval can result in longer switching durations. The values min *D*2 and max *D*2 are therefore not the same as the values *D*2min and *D*2max elsewhere in the present document.

C.2 Applying the timing model and related practical considerations

When designing an advert placement opportunity, the content provider needs to take into account the capabilities of terminals in terms of switching accuracy and durations.

In horizontal deployment scenarios it is expected that performance characteristics vary between devices. A broadcaster aiming to maximise reach of targeted advertising campaigns (or non-commercial substitution content) potentially needs to cater to terminals with the slowest and least accurate switching characteristics.

The limit of accuracy for the switch from broadcast to broadband defines a take-off period of the broadcast content (shown on Figure C.1). The content could be tailored such that it is acceptable to the viewer to stop presenting broadcast on any frame during this period, and so the minimum length of this period needs to be at least the switching accuracy limit *A1*.

A landing period of the broadcast is defined by the earliest and latest times at which the presentation of broadcast could resume.

If the broadband to broadcast switch time *T*2 is specified on the broadband content's timeline (as shown in Figure C.1), then:

- the landing period needs to begin at the earliest time at which presentation could resume which is
  $$\min D1 + L - \Delta + \min D2$$
  after timestamp *T*1;

- and the minimum length of this period is:
  $$A1 + A2 + 2\Delta + \max D1 - \min D1 + \max D2 - \min D2$$
  which is the difference between when the landing period ends of the slowest scenario:
  $$T1 + A1 + \max D1 + L + \Delta + A2 + \max D2$$
  and when it starts for the fastest scenario:
  $$T1 + \min D1 + L - \Delta + \min D2$$

If the broadband to broadcast switch time *T*2 is specified on the broadcast timeline (as shown in Figure YY) then:

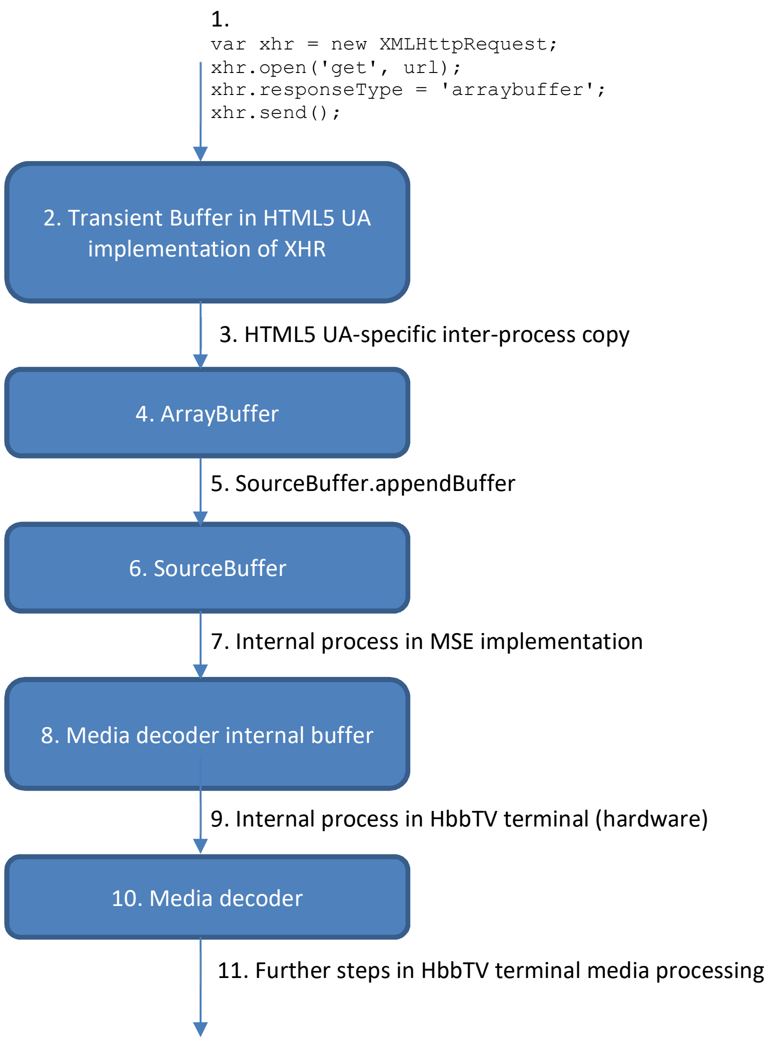
- the landing period needs to begin at timestamp
  $$\mathrm{T2} + \min D2$$
  and needs to have a minimum length of
  $$A2 + \max D2 - \min D2$$

If there are multiple consecutive substitutions, then the switching accuracy and duration at the transition from one substitution to the next also needs to be accounted for.

In all scenarios, the editorial content of the broadcast and substitution content needs to avoid requiring truly seamless continuity of audio and video during the switch. This is because, some broadcast audio content could be replaced by a brief moment of silence by the terminal, and some broadcast video content could be replaced by a brief period of black or freeze frame by the terminal.

Annex D (informative): Advert data flow, buffering and memory management

Figure D.1 illustrates an example of the data flow of an advert when played using MSE.


1.
2. Transient Buffer in HTML5 UA implementation of XHR
3. HTML5 UA-specific inter-process copy
4. ArrayBuffer
5. SourceBuffer.appendBuffer
6. SourceBuffer
7. Internal process in MSE implementation
8. Media decoder internal buffer
9. Internal process in HbbTV terminal (hardware)
10. Media decoder
11. Further steps in HbbTV terminal media processing

```
var xhr = new XMLHttpRequest;
xhr.open('get', url);
xhr.responseType = 'arraybuffer';
xhr.send();
```

Figure D.1: Example advert data flow with MSE

1) **xhr.send**: An HbbTV application will typically read the advert into the terminal using XMLHttpRequest

NOTE 1: Applications may get more predictable and inter-operable behaviour if they do multiple small XMLHttpRequests rather than attempting to load an entire advert in a single request as this may minimise the peak memory usage.

2) **Transient Buffer in HTML5 UA implementation of XHR**: In practice, HTML5 user agents typically read data into a transient buffer and not into the final destination.

3) **HTML5 UA-specific inter-process copy**: In practice HTML5 user agents typically perform network access in a different process from where the HTML5 rendering happens. (For one example of this see under “Which process controls what?" in the "Inside look at modern web browser (part 1)" [i.6]). Data is read in from the network into a transient buffer that is part of one process and then copied to the final destination.

4) **ArrayBuffer**: A JavaScript ArrayBuffer is the data type normally used for data to be passed to MSE. Note that the memory for a JavaScript ArrayBuffer comes from the general pool of memory available to the browser, whether there is enough memory available for a particular advert may change over time depending on what else is going on and what has happened previously. Also there is no standard way for an app to test in advance how much free memory is available to the browser.

5) **SourceBuffer.appendBuffer**: As defined in clause 3.2 of MSE [4]. Applications need to be written considering the possibility that the entire advert may not fit in the SourceBuffer in one operation.

NOTE 2: In order to minimise total memory usage, applications that are doing multiple XMLHttpRequests should consider discarding array buffers whose contents have been successfully appended to the corresponding SourceBuffer. Where there is a requirement for an entire advert to be in memory before it is played, it does not matter that some of the advert is in SourceBuffers and some of the advert is in ArrayBuffers.

6) **SourceBuffe**r: As defined in clause 3 of MSE [4]. The MSE specification [4] does not define minimum sizes for SourceBuffers. Implementations may start evicting data in a way that is implementation dependent. Information on existing practice can be found under "How much data can I append?" in "Exceeding the buffering quota" [i.7].

7) **Internal process in MSE implementation**: In practice there will typically be an implementation-specific intermediate buffer between the MSE SourceBuffer and the media decoder. The process by which media data is copied between the two is outside the scope of the present document.

8) **Media decoder internal buffer**: In practice there will typically be an implementation-specific intermediate buffer between the MSE SourceBuffer and the media decoder. Implementations sometimes do not start decoding media until this intermediate buffer has a certain amount of data in it. This may introduce a delay (latency) between when the first media data is appended to the SourceBuffer and when that media data is transferred to the media decoder.

9) **Internal process in HbbTV terminal (hardware)**: The process by which data is transferred from the media decoder internal buffer to the media decoder is implementation specific.

10) **Media decoder**: The (typically hardware) part of the HbbTV terminal that decodes the video and audio.

11) **Further steps in HbbTV terminal media processing**: The hardware used in HbbTV implementations may have media processing blocks immediately after the media decoders (i.e. before video and graphics are composed). It may also have media processing blocks between when video and graphics are composed and the panel (for a TV set) or HDMI output (for a STB).

Annex E (informative): Advert reporting

E.1 Introduction

HbbTV targeted advertising enables the delivery of digital video adverts over broadband that are then presented as part of linear TV services received by terminals via DVB-T(2)/C(2)/S(2) or IPTV. Thereby, digital video adverts are shown on targeted terminals instead of a broadcast advert, which will be seen by viewers on all non-targeted terminals. Although, non-commercial use cases are covered by HbbTV targeted advertising (i.e. non-commercial substitution content can be presented to viewers), this annex mainly addresses scenarios where there is a commercial intent for the substitution of broadcasted (advert) content by digital video adverts as it provides guidance on how advert reporting can be implemented by a DAS application.

Advert reporting is the provision of advert metrics to an advert server during or after the advert substitution process. Such advert metrics are captured in order to measure advert impressions, i.e. in order to be able to draw conclusions on whether or not a portion or even the whole digital video advert has been served and thus can be assumed to have been seen by viewers. Ad reporting metrics are created by using tracking elements defined in the VAST template (e.g. VAST 4.0 [i.9] and VAST 4.1 [i.10]) describing an advert or sequence of adverts to be rendered. Thereby, individual tracking elements are mapped to video events such as advert video start or completed. The reporting is mostly done from the client however it is recognised that there are also scenarios where some reporting metrics can be measured server-side. Advert reporting is understood to be a fundamental part of digital advertising as it allows to establish trust within the ecosystem.

HbbTV targeted advertising allows for the implementation of advert reporting solutions that support digital video advert metrics that are generally compliant with what is already well established and widely used in digital advertising (for example the IAB "Digital Video In-Stream Ad Metric Definitions" [i.12]).

E.2 Digital video adverts

In case of a single advert placement opportunity (i.e. for non-consecutive digital video adverts) it is possible to obtain the relevant data for player operation metrics and for linear advert metrics as defined in VAST 4.1 (see section 3.14.1 in [i.10]) and thus to track e.g. the corresponding events:

- *mute / unmute* (the audio of the digital video advert has been muted / unmuted by the viewer)
- *loaded* (indicates when player considers that it buffered the digital video advert fully or to the extent that it is ready to play the media)
- *start* (playback of a digital video advert began)
- *firstQuartile / midpoint / thirdQuartile* (the digital video advert was played at least for 25% / 50% / 75% of its duration at normal speed)
- *complete* (the digital video advert was played to the end at normal speed)
- *progress* (amount of video played at normal speed in units of time or as a percentage value if the DAS application allows a user to rewind, fast-forward or pause playback)

The following advert metrics are expected to be of relevance in addition to player operation metrics and linear advert metrics listed in clause 3.14.1 of VAST [i.10]:

- *subtitle* (the digital video advert has been shown with subtitles)
- *creativeView* (the technical version of a digital video advert that has been presented as explained under "view" in [i.12])

For a multiple advert placement opportunity (e.g. when a whole advert break is substituted) it is further recommended to report which adverts have been presented and in which order:

- *opportunityUse* (advert break is scheduled client-side by local stitching service vs. advert break is tailored and fully prepared server-side)

- *adSequence* (name / id of digital video adverts in order of playback or name / id of tailored advert break)

NOTE 1: If the advert metrics introduced for a single advert placement opportunity are not reported per single advert, these may be reported per advert break, but should then still be measured against each digital video advert.

The features and facilities needed for a DAS app to report on the advert metrics above are provided by TS 102 796 [1]. In particular, the HTML media element integration requirements in clause 9.6 and the key event requirements in clause 10.2.2.1 of TS 102 796 [1] enable a DAS app to determine the following:

- whether the current digital video advert can be played up to its end without being stopped due to further buffering of content (e.g. the `canplaythrough` event)
- whether playback of the current digital video advert has started
- whether playback of the current digital video advert has been stopped by the viewer
- whether playback of the current digital video advert has been paused / resumed by the viewer
- whether playback speed has changed during playback of the current digital video advert by the viewer
- the duration of the current digital video advert
- the playback position of the current digital video advert
- whether playback of the current digital video advert has reached the end
- whether subtitles are activated for the current digital video advert

Further, discovering whether audio is muted / unmuted for the current digital video advert can be done by reading the `LocalSystem.mute` property – see table A.1.

NOTE 2: Determining that playback of the current digital video advert has been discarded due to the viewer changing the channel may not be possible as this may terminate the DAS app immediately (see clause 6.2.2.2 in TS 102 796 [1]).

NOTE 3: Determining that playback of the current digital video advert has been discarded due to the viewer pressing the “Exit” button is not possible as this will terminate the DAS app immediately (see clause 6.2.2.3 in TS 102 796 [1]).

E.3 Interactive adverts

As interactive adverts are also in scope of HbbTV targeted advertising some additional nonlinear advert metrics may need to be supported. With respect to VAST 4.1 (see section 3.14.1 in [i.10]) these are e.g.:

- *acceptInvitation* (e.g. when a viewer decided to click on provided interactive advert components)

If the interactive component of an advert is accompanied by a display advert (i.e. an overlay) the following advert metrics may also be relevant:

- *impression* (when a display advert has been displayed)
- *adExpand* (when a display advert is set to full screen)
- *adCollapse* (when the viewer manually reduced the size of a display advert)
- *minimize* (when the viewer manually set a display advert to the smallest possible size)
- *close* (when the viewer manually closed a display advert)
- *overlayViewDuration* (the time that the display advert has been displayed)

It is recognised that all these interactive advert metrics can be implemented by a DAS app using standard features of TS

102 796 [1].

E.4 VAST error codes

VAST Error Codes are usually fired by the application when adverts cannot be served. This enables troubleshooting and optimisation of advert campaigns. VAST error codes as provided by the table in section 2.3.6.3 in [i.10] are considered to be supported by standard features of TS 102 796 [1].

E.5 Advert verification services

In order to build trust in advertising measurement it may be necessary to track parameters available to the DAS app, which give more information about how adverts are consumed (e.g. screen size and audio mute state). In addition, advert verification reporting may be delivered by a third party viewability provider. If required, this is expected to be implemented by means of a JavaScript library loaded at runtime. Consideration should be given by app developers to performance of such JavaScript libraries. It is noted that advert verification is supported by VAST 4.0/4.1 (see clause 1.2 in [i.9] and [i.10]) and that industry standardisation around advert verification metrics is being developed by the IAB Open Measurement Working Group https://iabtechlab.com/working-groups/open-measurement-working-group/).

Annex F (informative): Video and audio decoder resource allocation

F.1 Terminals with only one video and audio decoder available to HbbTV

In these terminals, the algorithm for attempting to allocate suitable video and audio decoders for `newMediaObject` will determine that the only suitable video and audio decoders for `newMediaObject` are those allocated to `originalMediaObject`. Step 2 of the fast media switch algorithm will stop `originalMediaObject` (ensuring that video and audio decoders are released) and then start `newMediaObject` which will allocate them.

F.2 Terminals with more than one video and audio decoder available to HbbTV – simple scenario

In these terminals, if the first switch is from broadcast to broadband, then the algorithm for attempting to allocate suitable video and audio decoders for `newMediaObject` will determine that a suitable video decoder and an audio decoder are not allocated to anything and can be allocated to `newMediaObject` when that algorithm is invoked as part of the call to the `switchMediaPresentation` method. After the switch, the video/broadcast object will retain the video and audio decoder that were previously allocated and will continue in the presenting state but either behind `newMediaObject` on the CSS z-axis or in front of it but with the CSS visibility set to `hidden`.

If the second switch would be from broadband back to broadcast then, when the `switchMediaPresentation` method is called, `newMediaObject` (being the video/broadcast object) will still be in the presenting state but with the CSS visibility set to `hidden`. The algorithm for attempting to allocate suitable video and audio decoders for `newMediaObject` will therefore have nothing to do and will stop at the first step. After the switch is over, the HTML5 video element that presented the broadband content will be in the paused state – either from having played the broadband content to the end or as a result of step 12) of the fast media switch algorithm. The video and audio decoder used by the video element will still be allocated to it but are allowed to be taken away when its in the paused state.

This scenario is illustrated by table F.1 below.

Table F.1: Simple scenario for dynamic advert substitution with multiple video and audio decoders using MSE

Media decoder	Before 1st advert	While 1st advert is playing	While 2nd advert is playing	While 3rd (last) advert is playing	After 3rd advert
#1	Broadcast				
#2	Available then initialised for MSE	MSE			Paused after MSE then available

Similar scenarios apply for dynamic ad substitution of a single advert using non-adaptive HTTP streaming or DASH however substituting multiple adverts using these technologies is more complex.

F.3 Terminals with more than one video and audio decoder available to HbbTV – more complex scenarios

The present document does not fully support more complex scenarios such as switching from broadcast to broadband, then from broadband to broadband and finally from broadband back to broadcast.

In practice it is likely that scenarios with an even number of broadband to broadband switches will work as these will result in the video and audio decoders originally used for the broadcast being available for the final switch back from broadband to broadcast. This is illustrated by table F.2 below.

Table F.2: Example dynamic advert substitution scenario with multiple video and audio decoders used alternately between multiple adverts – odd number of adverts

Media decoder	Before 1st advert	While 1st advert is playing	While 2nd advert is playing	While 3rd (last) advert is playing	After 3rd advert
#1	Broadcast	Broadcast (not visible) then initialised for 2nd advert	2nd advert	Paused after 2nd advert then initialised for broadcast	Broadcast
#2	Available then initialised for 1st advert	1st advert	Paused after 1st advert then initialised for 3rd advert	3rd advert	Paused after 3rd advert then available

In contrast, scenarios with an odd number of broadband / broadband switches will result in the video and audio decoder originally used for broadcast being used for the last piece of broadband delivered content as shown in table F.3 below. In this scenario, if media decoder #2 cannot be used for the broadcast for any reason then the final transition from broadband back to broadcast would involve stopping and re-starting media decoder #1 which risks a poor user experience if it has not been optimised. Here be dragons!

Table F.3: Example dynamic advert substitution scenario with multiple video and audio decoders used alternately between multiple adverts – even number of adverts

Media decoder	Before 1st advert	While 1st advert is playing	While 2nd advert is playing	While 3rd advert is playing	While 4th (last) advert is playing	After 4th advert
#1	Broadcast	Broadcast (not visible) then initialised for 2nd advert	2nd advert	Paused after 2nd advert then initialised for 4th advert	4th advert	??
#2	Available then initialised for 1st advert	1st advert	Paused after 1st advert then initialised for 3rd advert	3rd advert	Paused after 3rd advert	??

Of course some terminals may have more than two media decoders available in which case things are even more complex.

Annex (informative): Bibliography

•

Annex (informative): Change History

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

History

Document history		
<Version>	<Date>	<Milestone>

Latest changes made on 2018-02-09