Guidelines for Implementation: DASH-IF Interoperability Points

April 7, 2015

DASH Industry Forum

Version 3.0 (Final Version)

Scope

The scope of the interoperability points defined in this document is to provide support for high-quality video distribution for over the top services using H.264/AVC and H.265/HEVC. Both live and on-demand services are supported. The features address relevant use cases for live services, ad insertion, trick modes, content protection and subtitling. Extensions for multi-channel audio are defined.

Disclaimer

This is a document made available by DASH-IF. The technology embodied in this document may be covered under patents, including patents owned by such companies. No patent license, either implied or express, is granted to you by this document. This draft specification is provided on an as-is basis without any warranty whatsoever.

In addition, this document may include references to documents and/or technologies controlled by third parties. Those documents and technologies may be subject to third party rules and licensing terms. No intellectual property license, either implied or ex-press, to any third party material is granted to you by this document or DASH-IF. DASH-IF makes no any warranty whatsoever for such third party material.

Contents

List of Figures

List of Tables

Acronyms, abbreviations and definitions

For acronyms, abbreviations and definitions refer to ISO/IEC 23009-1 [4].

In addition, the following definitions are used in this document:

Ad Break: A location or point in time where one or more ads may be scheduled for delivery; same as *avail* and *placement opportunity*.

Ad Decision Service: functional entity that decides which ad(s) will be shown to the user. It interfaces deployment-specific and are out of scope for this document.

Ad Management Module: logical service that, given cue data, communicates with the ad decision service and determines which advertisement content (if at all) should be presented during the ad break described in the cue data.

Cue: indication of time and parameters of the upcoming ad break. Note that cues can indicate a pending switch to and ad break, pending switch to a next ad within an ad break, and pending switch from an ad break to the main content.

CDN node: functional entity returning a segment on request from DASH client. There are no assumptions on location of the node.

Packager: functional entity that processes conditioned content and produces media segments suitable for consumption by a DASH client. This entity is also known as fragmenter, encapsulater, or segmenter. Packager does not communicate directly with the server – its output is written to the origin.

Origin: functional entity that contains all media segments indicated in the MPD, and is the fallback if CDN nodes are unable to provide a cached version of the segment on client request.

Splice Point: point in media content where

MPD Generator: functional entity returning an MPD on request from DASH client. It may be generating an MPD on the fly or returning a cached one.

XLink resolver: functional entity which returns one or more remote elements on request from DASH client.

In addition, the following abbreviations and acronyms are used in this document:

AAC	Advanced Audio Coding
AVC	Advanced Video Coding
DRM	Digital Rights Management
DTV	Digital Television
FCC	Federal Communications Commission
GOP	Group-of-Pictures
HD	High-Definition

HDMI	High-Definition Multimedia Interface
HE-AAC	High Efficiency AAC
HEVC	High-Efficiency Video Coding
KID	common Key IDentifier
IDR	Instantaneous Decoder Refresh
MPEG	Moving Pictures Experts Group
PCM	Pulse Code Modulation
PPS	Picture Parameter Set
PS	Parametric Stereo
SBR	Spectral Band Replication
SD	Standard Definition
SEI	Supplemental Enhancement Information
SMPTE	Society of Motion Picture and Television Engineers
SPS	Sequence Parameter Set
TT	Timed Text
TTML	Timed Text Markup Language

References

[1] DASH-IF DASH-264/AVC Interoperability Points, version 1.0, available at http://dashif.org/w/2013/06/DASH-AVC-264-base-v1.03.pdf

[2] DASH-IF DASH-264/AVC Interoperability Points, version 2.0, available at http://dashif.org/w/2013/08/DASH-AVC-264-v2.00-hd-mca.pdf

[3] ISO/IEC 23009-1:2012/Cor.1:2013 Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats.

Note: this document is superseded by reference [4], but maintained as the initial version of this document is provided in the above reference.

[4] ISO/IEC 23009-1:2014/Cor.1:2015/Amd.1:2015 Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats.

Note: ISO/IEC 23009-1:2014/Amd.1:2015 is expected to be published shortly. The latest document is available in the MPEG output document w14860.

[5] ISO/IEC 23009-2:2015/Amd.3 Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats.

Note: ISO/IEC 23009-1:2014/Amd.3:2015 is expected to be published by end of 2015. The latest document is available in the MPEG output document w15219.

[6] ISO/IEC 23009-2:2014: Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 2: Conformance and Reference.

[7] ISO/IEC 23009-3:2014: Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 3: Implementation Guidelines (Study of ISO/IEC PDTR, available as w13514).

[8] ISO/IEC 14496-12:2015 Information technology -- Coding of audio-visual objects -- Part 12: ISO base media file format.

Note: ISO/IEC 14496-12:2015 is expected to be published shortly. The latest document is available in the MPEG output document w15177.

[9] ITU-T Recommendation H.264 (01/2012): "Advanced video coding for generic audiovisual services" | ISO/IEC 14496-10:2010: "Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding".

[10] ISO/IEC 14496-15:2014/Cor 1:2015: Information technology -- Coding of audio-visual objects -- Part 15: Carriage of network abstraction layer (NAL) unit structured video in ISO base media file format.

[11] IETF RFC 6381, The 'Codecs' and 'Profiles' Parameters for "Bucket" Media Types, August 2011.

[12] ISO/IEC 14496-3:2009 - Information technology -- Coding of audio-visual objects -- Part 3: Audio with Corrigendum 1:2009, Corrigendum 2:2011, Corrigendum 3:2012, Amendment 1:2009, Amendment 2:2010, Amendment 3:2012, and Amendment 4:2014.

[13] ISO/IEC 14496-14:2003/Amd 1:2010 Information technology -- Coding of audio-visual objects -- Part 14: The MP4 File Format

[14] 3GPP (2005-01-04). "ETSI TS 126 401 V6.1.0 (2004-12) - Universal Mobile Telecommunications System (UMTS); General audio codec audio processing functions; Enhanced aacPlus general audio codec; General description (3GPP TS 26.401 version 6.1.0 Release 6)"

[15] CEA-708-D: Digital Television (DTV) Closed Captioning, August 2008

[16] 3GPP TS 26.245: "Transparent end-to-end Packet switched Streaming Service (PSS); Timed text format"

[17] W3C Timed Text Markup Language (TTML) 1.0, November 2010.

[18] SMPTE ST 2052: "Timed Text"

[19] W3C WebVTT - W3C Web Video Text Tracks, Living Standard — http://dev.w3.org/html5/webvtt/

[20] ITU-T Recommendation H.265 (07/2013): "Advanced video coding for generic audiovisual services" | ISO/IEC 23008-2:2013: " High Efficiency Coding and Media Delivery in Heterogeneous Environments – Part 2: High Efficiency Video Coding", downloadable here: http://www.itu.int/rec/T-REC-H.265

[21] EBU Tech 3350, "EBU-TT, Part 1, Subtitling format definition", July 2012, http://tech.ebu.ch/docs/tech/tech3350.pdf?vers=1.0

[22] IETF RFC2616, Hypertext Transfer Protocol -- HTTP/1.1, June 1999.

Note: DASH-IF is aware of the recent updates of HTTP/1.1 in the IETF with the new RFCs 723[0..5]. The group is currently investigating the detailed required changes to update the references. A revised version 3.1 addressing this issue is expected to be published within a short time.

[23] Recommended Practice (Conversion from CEA 608 to SMPTE-TT) RP 2052-10-2012 https://www.smpte.org/sites/default/files/rp2052-10-2012.pdf

[24] Recommended Practice (Conversion from CEA 708 to SMPTE-TT) RP 2052-11-2013 https://www.smpte.org/sites/default/files/RP2052-11-2013.pdf

[25] ISO/IEC DIS 14496-30:2014, "Timed Text and Other Visual Overlays in ISO Base Media File Format".

[26] ISO/IEC 23001-7:2015: "Information technology -- MPEG systems technologies -- Part 7: Common encryption in ISO base media file format files".

Note: ISO/IEC 23001-7:2015 is expected to be published shortly. The latest document is available in the MPEG output document 14425.

[27] DASH Industry Forum, "Guidelines for Implementation: DASH-AVC/264 Test Cases and Vectors".

[28] DASH Industry Forum, "Guidelines for Implementation: DASH-AVC/264 Conformance Software", http://dashif.org/conformance.html.

[29] DASH Identifiers Repository, available here: http://dashif.org/identifiers

[30] DTS 9302J81100, “Implementation of DTS Audio in Media Files Based on ISO/IEC 14496”, http://www.dts.com/professionals/resources/resource-center.aspx

[31] ETSI TS 102 366 v1.2.1, Digital Audio Compression (AC-3, Enhanced AC-3) Standard (2008-08)

[32] MLP (Dolby TrueHD) streams within the ISO Base Media File Format, version 1.0, September 2009.

[33] ETSI TS 102 114 v1.3.1 (2011-08), “DTS Coherent Acoustics; Core and Extensions with Additional Profiles”

[34] ISO/IEC 23003-1:2007 - Information technology -- MPEG audio technologies -- Part 1: MPEG Surround

[35] DTS 9302K62400, “Implementation of DTS Audio in Dynamic Adaptive Streaming over HTTP (DASH)”, http://www.dts.com/professionals/resources/resource-center.aspx

[36] IETF RFC5905, "Network Time Protocol Version 4: Protocol and Algorithms Specification," June 2010.

[37] IETF RFC 6265: "HTTP State Management Mechanism", April 2011.

[38] DVB Document A168: "MPEG-DASH Profile for Transport of ISO BMFF Based DVB Services over IP Based Networks", July 2014, available here: https://www.dvb.org/resources/public/standards/a168_dvb-dash.pdf

[39] ANSI/SCTE 128-1 2013: " AVC Video Constraints for Cable Television, Part 1 - Coding", available here: http://www.scte.org/documents/pdf/Standards/ANSI_SCTE%20128-1%202013.pdf

[40] IETF RFC 2119, "Key words for use in RFCs to Indicate Requirement Levels", April 1997.

[41] ISO: “ISO 639.2, Code for the Representation of Names of Languages — Part 2: alpha-3 code,” as maintained by the ISO 639/Joint Advisory Committee (ISO 639/JAC), http://www.loc.gov/standards/iso639-2/iso639jac.html; JAC home page: http://www.loc.gov/standards/iso639-2/iso639jac.html; ISO 639.2 standard online: http://www.loc.gov/standards/iso639-2/langhome.html.

[42] CEA-608-E, Line 21 Data Service, March 2008.

[43] IETF RFC 5234, “Augmented BNF for Syntax Specifications: ABNF”, January 2008.

[44] SMPTE ST 2086:2014, “Mastering Display Color Volume Metadata Supporting High Luminance And Wide Color Gamut Images”

[45] ISO/IEC 23001-8:2013, “Information technology -- MPEG systems technologies -- Part 8: Coding-independent code points”, available here: http://standards.iso.org/ittf/PubliclyAvailableStandards/c062088_ISO_IEC_23001-8_2013.zip

[46] IETF RFC 7164, “RTP and Leap Seconds”, March 2014

[47] DASH Industry Forum, "Guidelines for Implementation: DASH264 Interoperability 3 Points", http://dashif.org/w/2013/08/DASH-AVC-264-v2.00-hd-mca.pdf .

[48] IAB Video Multiple Ad Playlist (VMAP), http://www.iab.net/media/file/VMAPv1.0.pdf

[49] IAB Video Ad Serving Template (VAST), http://www.iab.net/media/file/VASTv3.0.pdf

[50] ANSI/SCTE 35 2014, Digital Program Insertion Cueing Message for Cable

[51] ANSI/SCTE 67 2014, Recommended Practice for SCTE 35 Digital Program Insertion Cueing Message for Cable

[52] SCTE DVS 1202, MPEG DASH for IP-Based Cable Services, Part 1: MPD Constraints and Extensions

[53] SCTE DVS 1208, MPEG DASH for IP-Based Cable Services, Part 3: DASH/FF Profile

Note: SCTE DVS 1202 and 1208 are expected to be published shortly as ANSI/SCTE standards.

[54] EIDR ID Format - EIDR_ID_Format_v1.02_Jan2012-1.pdf, www.eidr.org

[55] Common Metadata, TR-META-CM, ver. 2.0, January 3, 2013, available at http://www.movielabs.com/md/md/v2.0/Common_Metadata_v2.0.pdf

[56] IETF RFC 4648, "The Base16, Base32, and Base64 Data Encodings", October 2006.

1. Introduction

This document defines DASH-IF's interoperability points. The document includes interoperability points for only this version of the document. For earlier versions, please refer to version 1 [1] and version 2 of this document. DASH-IF recommends to deprecate the Interoperability Points in previous versions and deploy using one of the Interoperability Points and extensions in this document.

As a historical note, the scope of the initial DASH-AVC/264 interoperability point as issued in version 1 of this document [1] was the basic support high-quality video distribution over the top. Both live and on-demand services are supported.

In the second version of this document [2], HD video (up to 1080p) extensions and several multichannel audio extensions are defined.

In this third version this document, only two DASH-264/AVC IOP are defined as well as two extensions to support HEVC [20] are defined. Detailed refinements and improvements for DASH-IF live services and for ad insertion were added. The main differentiations are, that one IOP adds additional requirements on the client to support segment parsing.

This document defines the Interoperability Points as documented in Table 1 and Extensions as defined in Table 2. The version in which each Interoperability Point and Extension was added is also provided in the tables.

Note that all version 1 IOPs are also defined in version 2 and therefore referencing version [2] is considered sufficient.

Table 1 DASH-IF Interoperability Points

Interoperability Point	Identifier	Version	Reference
DASH-AVC/264	`http://dashif.org/guidelines/dash264`	1.0	[2], 6.3
DASH-AVC/264 SD	`http://dashif.org/guidelines/dash264#sd`	1.0	[2], 7.3
DASH-AVC/264 HD	`http://dashif.org/guidelines/dash264#hd`	2.0	[2], 8.3
DASH-AVC/264 main	`http://dashif.org/guidelines/dash264main`	3.0	8.2
DASH-AVC/264 high	`http://dashif.org/guidelines/dash264high`	3.0	0
DASH-IF IOP simple	`http://dashif.org/guidelines/dash-if-simple`	3.0	8.4
DASH-IF IOP main	`http://dashif.org/guidelines/dash-if-main`	3.0	8.5

Note that all extensions defined in version 2 of this document are carried over into version 3.0 without any modifications. In order to maintain a single document, referencing in Table 2 is restricted to this document.

Table 2 DASH-IF Interoperability Point Extensions

Extension	Identifier	Version	Section
DASH-IF multichannel audio extension with Enhanced AC-3	`http://dashif.org/guidelines/dashif#ec-3`	2.0	9.4.2.3
DASH-IF multichannel extension with Dolby TrueHD	`http://dashif.org/guidelines/dashif#mlpa`	2.0	9.4.2.3
DASH-IF multichannel audio extension with DTS Digital Surround	`http://dashif.org/guidelines/dashif#dtsc`	2.0	9.4.3.3
DASH-IF multichannel audio extension with DTS-HD High Resolution and DTS-HD Master Audio	`http://dashif.org/guidelines/dashif#dtsh`	2.0	9.4.3.3
DASH-IF multichannel audio extension with DTS Express	`http://dashif.org/guidelines/dashif#dtse`	2.0	9.4.3.3
DASH-IF multichannel extension with DTS-HD Lossless (no core)	`http://dashif.org/guidelines/dashif#dtsl`	2.0	9.4.3.3
DASH-IF multichannel audio extension with MPEG Surround	`http://dashif.org/guidelines/dashif#mps`	2.0	9.4.4.3
DASH-IF multichannel audio extension with HE-AACv2 level 4	`http://dashif.org/guidelines/dashif#heaac-mc51`	2.0	9.4.5.3
DASH-IF multichannel audio extension with HE-AACv2 level 6	`http://dashif.org/guidelines/dashif#heaac-mc71`	2.0	9.4.5.3

Test cases and test vectors for DASH-AVC/264 Interoperability Points are provided in [27]. The conformance and reference software for DASH-AVC/264 Interoperability Points is provided in [28] (based on the MPEG conformance software [6]). DASH Identifiers for different categories can be found online [29].

2. Context and Conventions

2.1. Relation to MPEG-DASH and other DASH specifications

Dynamic Adaptive Streaming over HTTP (DASH) is initially defined in the first edition of ISO/IEC 23009-1 which was published in April 2012 and some corrections were done in 2013 [1].

In May 2014, ISO/IEC published the second version of ISO/IEC 23009-1 [5] that includes additional features and provide additional clarifications. The initial two versions of this document where based on the first edition of ISO/IEC 23009-1. This version is based on the second edition of ISO/IEC 23009-1, i.e. ISO/IEC 23009-1:2014 including Cor.1 and Amd.1 [5]. This means that also for all interoperability points that were initially defined in earlier versions of the document, also now the second edition serves as the reference. Backward-compatibility across different edition is handled by MPEG-DASH in ISO/IEC 23009-1 [5].

This document was generated in close coordination with DVB-DASH [38]. The tools and features are aligned to the extent considered reasonable. To support implementers, this document attempts to highlight any differences and/or further restrictions or extensions when compared to DVB-DASH. However, as a disclaimer, this coverage is not considered complete.

2.2. Compatibility and Extensions to Earlier Versions

2.2.1. Summary of Version 3 Modifications

Version 3 of this document applies the following modifications compared to version 2 [2]:

- Reference to the second edition of ISO/IEC 23009-1 including amendment 1 and cor.1 [4], as well as well as Amendment 3 [5].
- Add an explicit statement in DASH-264/AVC to forbid time code wrap around
- Definition on the usage of key words in section 2.3.
- Add more constraints on the usage of Trick Modes for improved interoperability in section 3.2.9.
- Add more constraints on the Representations in one Adaptation Set in section 3.2.10, especially when the bitstream switching is true.
- Add additional details on the usage of HTTP in section 3.4.
- Add H.265/HEVC as a codec and create IOPs for inclusion of this codec.
- Add CEA-608/708 closed captioning in SEI messages in section 6.4.3.
- Detailed description of simple and main live operation, with the latter including segment parsing in section 4.
- Detailed description of server-based and app-based ad insertion in section 5
- General editorial updates and clarifications
- Updates and clarification to section 7 on DRM and common encryption.
- Update to references
- Relaxation of the audio encoding requirements in section 6.3.2.
- Add clarification on the usage of the minimum buffer time and bandwidth in section 3.2.8.
- Add an informative section on the timing model of DASH in section 3.2.7.
- Relax the use of the '`lmsg`' brand for signaling the last segment in section 3.6.

- Simplification of the codecs table

2.2.2. Backward-Compatibility Considerations

Generally, content can be offered such that it can be consumed by version 2 and version 3 clients. In such a case the restricted authoring should be used and it should be accepted that version 2 clients may ignore certain Representations and Adaptation Sets. Content Authors may also consider the publication of two MPDs, but use the same segment formats.

In terms of compatibility between version 2 and version 3, the following should be considered:

- The backward-compatibility across MPEG editions is handled in the second edition of ISO/IEC 23009-1 [5].
- General clarifications and updates are added
- Further restrictions on content authoring compared to version 2 are:
    - forbid time code wrap around
    - the usage of DRM, especially the Content Protection element
    - constraints on trick mode usage
    - additional constraints on the usage of HTTP
    - Adaptation Set constraints
- Relaxations are:
    - Permit usage of additional subtitling format based on CEA-608/708
    - the audio encoding requirements for HE-AACv2
    - the use of the '`lmsg`' brand for signaling the last segment
    - the ability to signal bitstream switching set to true
    - the use of remote elements with Xlink

2.3. Use of Key Words

2.3.1. Background

DASH-IF generally does not write specifications, but provides and documents guidelines for implementers to refer to interoperability descriptions. In doing so, the DASH-IF agreed to use key words in order to support readers of the DASH-IF documents to understand better how to interpret the language. The usage of key words in this document is provided below.

2.3.2. Key Words

The key word usage is aligned with the definitions in RFC 2119 [40], namely:

- SHALL: This word means that the definition is an absolute requirement of the specification.

- SHALL NOT This phrase means that the definition is an absolute prohibition of the specification.
- SHOULD This word means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- SHOULD NOT This phrase means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- MAY: This word means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item.

These key words are attempted to be used consistently in this document, but only in small letters.

2.3.3. Mapping to DASH-IF Assets

If an IOP document associates such a key word from above to a content authoring statement then the following applies:

- SHALL: The conformance software provides a conformance check for this and issues an *error* if the conformance is not fulfilled.
- SHALL NOT: The conformance software provides a conformance check for this and issues an *error* if the conformance is not fulfilled.
- SHOULD: The conformance software provides a conformance check for this and issues a *warning* if the conformance is not fulfilled.
- SHOULD NOT: The conformance software provides a conformance check for this and issues a *warning* if the conformance is not fulfilled.
- SHOULD and MAY: If present, the feature check of the conformance software documents a feature of the content.

If an IOP document associates such a key word from above to a DASH Client then the following applies:

- SHALL: Test content is necessarily provided with this rule and the reference client implements the feature.
- SHALL NOT: The reference client does not implement the feature.
- SHOULD: Test content is provided with this rule and the reference client implements the feature unless there is a justification for not implementing this.
- SHOULD NOT: The reference client does not implement the feature unless there is a justification for implementing this.
- MAY: Test content is provided and the reference client implements the feature if there is a justification this.

2.4. Definition and Usage of Interoperability Points

2.4.1. Profile Definition in ISO/IEC 23009-1

MPEG DASH defines formats for MPDs and Segments. In addition MPEG provides the ability to further restrict the applied formats by the definition of *Profiles* as defined on section 8 of ISO/IEC 23009-1 [5]. Profiles of DASH are defined to enable interoperability and the signaling of the use of features.

Such a profile can also be understood as permission for DASH clients that implement the features required by the profile to process the Media Presentation (MPD document and Segments).

Furthermore, ISO/IEC 23009-1 permits external organizations or individuals to define restrictions, permissions and extensions by using this profile mechanism. It is recommended that such external definitions be not referred to as profiles, but as *Interoperability Points*. Such an interoperability point may be signalled in the `@profiles` parameter once a URI is defined. The owner of the URI is responsible to provide sufficient semantics on the restrictions and permission of this interoperability point.

This document makes use of this feature and provides a set of Interoperability Points. Therefore, based on the interoperability point definition, this document may be understood in two ways:

- a collection of content conforming points, i.e. as long as the content conforms to the restrictions as specified by the IOP, clients implementing the features can consume the content.
- a client capability points that enable content and service providers for flexible service provisioning to clients conforming to these client capabilities.

This document provides explicit requirements, recommendations and guidelines for content authoring that claims conformance to a profile (by adding the `@profiles` attribute to the MPD) as well as for clients that are permitted to consume a media presentation that contains such a profile.

2.4.2. Usage of Profiles

A Media Presentation may conform to one or multiple profiles/interoperability points and conforms to each of the profiles indicated in the **`MPD`**`@profiles` attribute is specified as follows:

When `ProfA` is included in the **`MPD`**`@profiles` attribute, the MPD is modified into a profile-specific MPD for profile conformance checking using the following ordered steps:

1. The **`MPD`**`@profiles` attribute of the profile-specific MPD contains only `ProfA`.
2. An **`AdaptationSet`** element for which `@profiles` does not or is not inferred to include `ProfA` is removed from the profile-specific MPD.
3. A Representation element for which `@profiles` does not or is not inferred to include `ProfA` is removed from the profile-specific MPD.
4. All elements or attributes that are either (i) in this Part of ISO/IEC 23009 and explicitly excluded by `ProfA`, or (ii) in an extension namespace and not explicitly included by `ProfA`, are removed from the profile-specific MPD.

5. All elements and attributes that “may be ignored” according to the specification of `ProfA` are removed from the profile-specific MPD.

An MPD is conforming to profile `ProfA` when it satisfies the following:

1. `ProfA` is included in the **`MPD`**`@profiles` attribute.
2. The profile-specific MPD for `ProfA` conforms to ISO/IEC 23009-1
3. The profile-specific MPD for `ProfA` conforms to the restrictions specified for `ProfA`.

A Media Presentation is conforming to profile `ProfA` when it satisfies the following:

1. The MPD of the Media Presentation is conforming to profile `ProfA` as specified above.
2. There is at least one Representation in each Period in the profile-specific MPD for `ProfA`.
3. The Segments of the Representations of the profile-specific MPD for `ProfA` conform to the restrictions specified for `ProfA`.

2.4.3. Interoperability Points and Extensions

This document defines Interoperability Points and Extensions. Both concepts make use of the profile functionality of ISO/IEC 23009-1.

Interoperability Points provide a basic collection of tools and features to ensure that content/service providers and client vendors can rely to support a sufficiently good audio-visual experience. Extensions enable content/service providers and client vendors to enhance the audio-visual experience provided by an Interoperability Point in a conforming manner.

The only difference between Interoperability Points and Extensions is that Interoperability Points define a full audio-visual experience and Extensions enhance the audio-visual experience in typically only one dimension.

Examples for the usage of the `@profiles` signaling are provided in Annex A of this document.

3. DASH-Related Aspects

3.1. Scope

DASH-IF Interoperability Points use ISO base media file format [8] based encapsulation and provide significant commonality with a superset of the ISO BMFF On-Demand and the ISO BMFF Live profile as defined in ISO/IEC 23009-1 [4], sections 8.3 and 8.4, respectively. DASH-IF IOPs are intended to provide support for on-demand and live content. The primary constraints imposed by this profile are the requirement that each Representation is provided in one of the following two ways

- as a single Segment, where Subsegments are aligned across Representations within an Adaptation Set. This permits scalable and efficient use of HTTP servers and simplifies seamless switching. This is mainly for on-demand use cases.
- as a sequence of Segments where each Segment is addressable by a template-generated URL. Content generated in this way is mainly suitable for dynamic and live services.

In both cases (Sub)Segments begin with Stream Access Points (SAPs) of type 1 or 2 [8], i.e. regular IDR frames in case of video. In addition, (Sub)Segments are constrained so that for switching video Representations within one Adaptation Set the boundaries are aligned without gaps or overlaps in the media data. Furthermore, switching is possible by a DASH client that downloads, decodes and presents the media stream of the come-from Representation and then switches to the go-to Representation by downloading, decoding and presenting the new media stream. No overlap in downloading, decoding and presentation is required for seamless switching of Representations in one Adaptation Set.

Additional constraints are documented for bitstream switching set to true as well as special case such as trick modes, etc.

3.2. DASH Formats

3.2.1. Introduction

This section introduces the detailed constraints of the MPD and the DASH segments in a descriptive way referring to ISO/IEC 23009-1 [4]. The DASH-based restrictions have significant commonality with the ISO BMFF Live and On-Demand profiles from the MPEG-DASH specification.

Specifically:

- Segment formats are based on ISO BMFF with fragmented movie files, i.e. (Sub)Segments are encoded as movie fragments containing a track fragment as defined in ISO/IEC 14496-12 [8], plus the following constraints to make each movie fragment independently decodable:
  - Default parameters and flags shall be stored in movie fragments (‘`tfhd`’ or ‘`trun`’ box) and not track headers (‘`trex`’ box)
  - The ‘`moof`’ boxes shall not use external data references, the flag ‘`default-base-is-moof`’ shall also be set (aka movie-fragment relative addressing) and `data-offset` shall be used, i.e. `base-data-offset-present` shall not be used (follows ISO/IEC 23009-1 [4]).
- Alignment with ISO BMFF Live & On-Demand Profiles, i.e. within each Adaptation Set the following applies
  - Fragmented movie files are used for encapsulation of media data
  - (Sub)Segments are aligned to enable seamless switching

Beyond the constraints provided in the ISO BMFF profiles, the following additional restrictions are applied.

- IDR-like SAPs (i.e., SAPs type 2 or below) at the start of each (Sub)Segment for simple switching.
- Segments should have almost equal duration. The maximum tolerance of segment duration shall be ±50% and the maximum accumulated deviation over multiple segments shall be ±50% of the signaled segment duration (i.e. the `@duration`). Such fluctuations in actual segment duration may be caused by for example ad replacement or specific IDR frame

placement. Note that the last segment in a Representation may be shorter according to ISO/IEC 23009-1 [4].

> Note: If accurate seeking to specific time is required and at the same time a fast response is required one may use On-Demand profile for VoD or the **`SegmentTimeline`** based addressing. Otherwise the offset in segment duration compared to the actual media segment duration may result in a less accurate seek position for the download request, resulting in some increased initial start-up.

- If the **`SegmentTimeline`** element is used for the signaling of the Segment duration, the timing in the segment timeline shall be media time accurate and no constraints on segment duration deviation are added except the maximum segment duration as specified in the MPD. However, despite the usage of the the **`SegmentTimeline`**, it is not encouraged to use varying Segment durations. The **`SegmentTimeline`** element should only be used in order to signal occasional shorter Segments (possibly caused by encoder processes) or to signal gaps in the time line.
- only non-multiplexed Representations shall be used, i.e. each Representation only contains a single media component.
- Addressing schemes are restricted to
  - templates with number-based addressing
  - templates with time-based addressing
  - Subsegments with Segment Index. In this case either the `@indexRange` attribute shall be present or the **`RepresentationIndex`** element shall be present. Only a single `sidx` box shall be present.

> Note 1: the external Representation Index was only added in the second edition [4]. If compatibility to v2.0 or earlier of this document is necessary, the external Representation Index shall not be used.

> Note 2: The latter restriction was introduced in version 3 of this document based on deployment experience and to enable alignment with DVB DASH.

- In case multiple Adaptation Sets with `@contentType='video'` are offered, exactly one video Adaptation Set shall be signaled as the main one unless different Adaptation Sets contain the same content with different quality or different codecs. In the latter case, all Adaptation Sets with the same content shall be signaled as the main content. Signalling as main content shall be done by using the Role descriptor with `@schemeIdUri="urn:mpeg:dash:role:2011"` and `@value="main"`.
- The content offering shall adhere to the presence rules of certain elements and attributes as defined section 3.2.4.

It is expected that a client conforming to such a profile is able to process content offered under these constraints. More details on client procedures are provided in section 3.3.

3.2.2. Media Presentation Description constraints for v1 & v2 Clients

3.2.2.1. Definition according to ISO/IEC 23009-1

This section follows a description according to ISO/IEC 23009-1. In section 3.2.2.2, a restricted content offering is provided that provides a conforming offering.

> NOTE: The term "ignored" in the following description means, that if an MPD is provided and a client that complies with this interoperability point removes the element that may be ignored, then the MPD is still complying with the constraints of the MPD and segments as defined in ISO/IEC 23001-9, section 7.3.

The MPD shall conform to the ISO Base Media File Format Common profile as defined on ISO/IEC 23009-1:2014/Amd.1:2015 [4], section 8.9, except for the following issues:

- Representations with `@mimeType` attribute `application/xml+ttml` shall not be ignored.

In addition, the Media Presentation Description shall conform to the following constraints:

- **`Representation`** elements with a `@subsegmentStartsWithSAP` value set to 3 may be ignored.
- **`Representation`** elements with a `@startsWithSAP` value set to 3 may be ignored.
- If a Period contains multiple Adaptation Sets with `@contentType="video"` then at least one Adaptation Set shall contain a Role element `<Role schemeIdUri="urn:mpeg:dash:role:2011" value="main">` and each Adaptation Set containing such a **`Role`** element shall provide perceptually equivalent media streams.

3.2.2.2. Simple Restricted Content Offering

A conforming MPD offering based on the ISO BMFF Live Profile shall contain

- **`MPD`**`@type` set to `static` or set to `dynamic`.
- **`MPD`**`@profiles` includes `urn:mpeg:dash:profile:isoff-live:2011`
- One or multiple Periods with each containing one or multiple Adaptation Sets and with each containing one or multiple Representations.
- The Representations contain or inherit a **`SegmentTemplate`** with `$Number$` or `$Time$` Identifier.
- `@segmentAlignment` set to true for all Adaptation Sets

A conforming MPD offering based on the ISO BMFF On-Demand Profile shall contain

- **`MPD`**`@type` set to `static`.
- **`MPD`**`@profiles` includes `urn:mpeg:dash:profile:isoff-ondemand:2011`

- One or multiple Periods with each containing one or multiple Adaptation Sets and with each containing one or multiple Representations.
- `@subSegmentAlignment` set to true for all Adaptation Sets

3.2.3. Segment format constraints

Representations and Segments referred to by the Representations in the profile-specific MPD for this profile, the following constraints shall be met:

— Representations shall comply with the formats defined in ISO/IEC 23009-1, section 7.3.

— In Media Segments, all Segment Index ('`sidx`') and Subsegment Index ('`ssix`') boxes, if present, shall be placed before any Movie Fragment ('`moof`') boxes.

Note: DVB DASH [38] permits only one single Segment Index box ('sidx') for the entire Segment. As this constraints is not severe in the content offering, it is strongly recommended to offer content following this constraint.

— If the **MPD**`@type` is equal to "`static`" and the **MPD**`@profiles` attribute includes "`urn:mpeg:dash:profile:isoff-on-demand:2011`", then

— Each Representation shall have one Segment that complies with the Indexed Self-Initializing Media Segment as defined in section 6.3.5.2 in ISO/IEC 23009-1.

— Time Codes expressing presentation and decode times shall be linearly increasing with increasing Segment number in one Representation. In order to minimize the frequency of time code wrap around 64 bit codes may be used or the timescale of the Representation may be chosen as small as possible. In order to support time code wrap around, a new Period may be added in the MPD added that initiates a new Period in order to express a discontinuity.

3.2.4. Presence of Attributes and Elements

Elements and attributes are expected to be present for certain Adaptation Sets and Representations to enable suitable initial selection and switching.

Specifically the following applies:

- For any Adaptation Sets with `@contentType="video"` the following attributes shall be present
  - o `@maxWidth` (or `@width` if all Representations have the same width)
  - o `@maxHeight` (or `@height` if all Representations have the same height)
  - o `@maxFrameRate` (or `@frameRate` if all Representations have the same frame rate)
  - o `@par`

Note: The attributes `@maxWidth` and `@maxHeight` should be used such that they describe the target display size. This means that they may exceed the actual largest size of any coded Representation in one Adaptation Set.

- For any Representation within an Adaptation Set with `@contentType="video"` the following attributes shall be present:
    - `@width`, if not present in **`AdaptationSet`** element
    - `@height`, if not present in **`AdaptationSet`** element
    - `@frameRate`, if not present in **`AdaptationSet`** element
    - `@sar`

    Note: `@width, @height`, and `@sar` attributes should indicate the vertical and horizontal sample count of encoded and cropped video samples, not the intended display size in pixels.
- For Adaptation Set or for any Representation within an Adaptation Set with `@contentType="video"` the attribute `@scanType` shall either not be present or shall be set to "`progressive`".
- For any Adaptation Sets with value of the `@contentType="audio"` the following attributes shall be present
    - `@lang`
- For any Representation within an Adaptation Set with value of the `@contentType="audio"` the following elements and attributes shall be present:
    - `@audioSamplingRate`, if not present in **`AdaptationSet`** element
    - **`AudioChannelConfiguration`**, if not present in **`AdaptationSet`** element

3.2.5. MPD Dimension Constraints

No constraints are defined on MPD size, or on the number of elements. However, it should be avoided to create unnecessary large MPDs.

Note: DVB DASH [38] adds MPD dimension constraints in section 4.5 of their specification. In order to conform to this specification, it is recommended to obey these constraints.

3.2.6. Generic Metadata

Generic metadata may be added to MPDs based on DASH. For this purpose, the Essential Property Descriptor and the Supplemental Property Descriptor as defined in ISO/IEC 23009-1 [4], clause 5.8.4.7 and 5.8.4.8.

Metadata identifiers for content properties are provided here: `http://dashif.org/identifiers`.

However, it is not expected that DASH-IF clients support all metadata at `http://dashif.org/identifiers` unless explicitly required.

3.2.7. DASH Timing Model

3.2.7.1. General

According to ISO/IEC 23009-1, DASH defines different timelines. One of the key features in DASH is that encoded versions of different media content components share a common timeline. The presentation time of each access unit within the media content is mapped to the global common presentation timeline for synchronization of different media components and to enable seamless switching of different coded versions of the same media components. This timeline is referred as Media Presentation timeline. The Media Segments themselves contain accurate Media Presentation timing information enabling synchronization of components and seamless switching.

A second timeline is used to signal to clients the availability time of Segments at the specified HTTP-URLs. These times are referred to as **Segment availability times** and are provided in wall-clock time. Clients typically compare the wall-clock time to Segment availability times before accessing the Segments at the specified HTTP-URLs in order to avoid erroneous HTTP request responses. For static Media Presentations, the availability times of all Segments are identical. For dynamic Media Presentations, the availability times of segments depend on the position of the Segment in the Media Presentation timeline, i.e. the Segments get available over time.

Figure 1 provides an overview of the different timelines in DASH and their relation. The diagram shows three Periods, each of the Periods contains multiple Representations (for the discussion it is irrelevant whether these are included in the same Adaptation Set or in different ones).

Specifically, the following information is available in the MPD that relates to timing:

- **`MPD`**`@availabilityStartTime`: the start time is the anchor for the MPD in wall-clock time. The value is denoted as *AST*.
- **`Period`**`@start`: the start time of the Period relative to the MPD availability start time. The value is denoted as *PS*.
- **`Representation`**`@presentationTimeOffset`: the presentation time offset of the Representation in the Period, i.e. it provides the time of the presentation that is supposed to be rendered at the start of the Period. Note that typically this time is either earliest presentation time of the first segment or a value slightly larger in order to ensure synchronization. If larger, this Representation is presented with short delay with respect to the Period start.

In addition, with the use of the **`Representation`**`@duration` or **`Representation.SegmentTimeline`** the MPD start time for each segment and the MPD duration for each segment can be derived. For details refer to ISO/IEC 23009-1.

According to Figure 1, the *AST* is a wall-clock time. It provides an anchor to all wall-clock time computation in the MPD. The sum of the **`Period`**`@start` of the first Period and the *AST* provides the *PST1* value in wall-clock time of the first Period. Each Representation is assigned a presentation time offset, either by the value of the attribute **`Representation`**`@presentationTimeOffset` or by default set to 0. The value of this attribute is denoted as PTO.

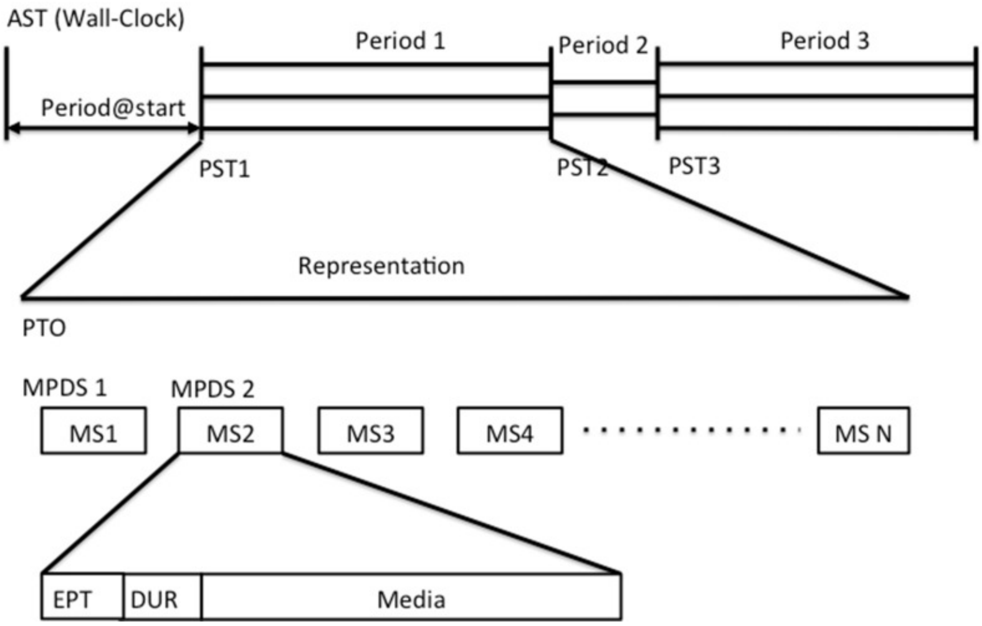


Figure 1 Overview Timing Model

Within a Representation, each segment is assigned an MPD start time and MPD duration according to ISO/IEC 23009-1. These two values can be computed from the MPD and provide approximate times for each segment that are in particular useful for random access and seeking.

In addition, each segment has an internal sample-accurate presentation time. Therefore, each segment has a media internal earliest presentation time *EPT* and sample-accurate presentation duration *DUR*.

For each media segment in each Representation the MPD start time of the segment should approximately be *EPT* - *PTO*. Specifically, the MPD start time shall be in the range of *EPT* - *PTO* - *0.5*DUR* and *EPT* - *PTO* + *0.5*DUR* according to the requirement stated above.

Each Period is treated independently. Details on processing at Period boundaries are provided in ISO/IEC 23009-1. One example is, that for time code wrap-around a new Period is added, that restarts at presentation time 0.

3.2.7.2. Static Media Presentations

For static media presentations, all Segments shall be available at time *AST*. This means that a DASH client may use the information in the MPD in order to seek to approximate times.

3.2.7.3. Dynamic Media Presentations

For dynamic media presentations, segments get available over time. The latest time they shall be available is at the sum of *PST* (which is *AST* + **`Period`**`@start`), MPD start time and MPD duration. The latter is added in order to take into account that at the server a segment typically needs to be completed prior to its availability.

3.2.8. Bandwidth and Minimum Buffer Time

The MPD contains a pair of values for a bandwidth and buffering description, namely the Minimum Buffer Time (`MBT`) expressed by the value of **`MPD`**`@minBufferTime` and bandwidth (`BW`) expressed by the value of **`Representation`**`@bandwidth`. The following holds:

- the value of the minimum buffer time does not provide any instructions to the client on how long to buffer the media. The value however describes how much buffer a client should have under ***ideal*** network conditions. As such, MBT is not describing the burstiness or jitter in the network, it is describing the burstiness or jitter in the **content encoding**. Together with the `BW` value, it is a property of the content. Using the "leaky bucket" model, it is the size of the bucket that makes `BW` true, given the way the content is encoded.
- The minimum buffer time provides information that for each Stream Access Point (and in the case of DASH-IF therefore each start of the Media Segment), the property of the stream: If the Representation (starting at any segment) is delivered over a constant bitrate channel with bitrate equal to value of the BW attribute then each presentation time *PT* is available at the client latest at time with a delay of at most *PT* + *MBT*.
- In the absence of any other guidance, **the** MBT **should be set** to the maximum GOP size (coded video sequence) of the content, which quite often is identical **to the maximum segment duration** for the live profile or the **maximum subsegment duration** for the On-Demand profile. The *MBT* may be set to a smaller value than maximum (sub)segment duration, but should not be set to a higher value.

In a simple and straightforward implementation, a DASH client decides downloading the next segment based on the following status information:

- the currently available buffer in the media pipeline, *buffer*
- the currently estimated download rate, *rate*
- the value of the attribute `@minBufferTime`, *MBT*
- the set of values of the `@bandwidth` attribute for each Representation i, *BW*[*i*]

The task of the client is to select a suitable Representation *i*.

The relevant issue is that starting from a SAP on, the DASH client can continue to playout the data. This means that at the current time it does have *buffer* data in the buffer. Based on this model the client can download a Representation *i* for which $BW[i] \leq rate*buffer/MBT$ without emptying the buffer.

Note that in this model, some idealizations typically do not hold in practice, such as constant bitrate channel, progressive download and playout of Segments, no blocking and congestion of other HTTP requests, etc. Therefore, a DASH client should use these values with care to compensate such practical circumstances; especially variations in download speed, latency, jitter, scheduling of requests of media components, as well as to address other practical circumstances.

One example is if the DASH client operates on Segment granularity. As in this case, not only parts of the Segment (i.e., `MBT`) needs to be downloaded, but the entire Segment, and if the `MBT` is smaller than the Segment duration, then rather the segment duration needs to be used instead of the MBT for the required buffer size and the download scheduling, i.e. download a Representation *i* for which $BW[i] \leq rate*buffer/max\_segment\_duration$.

3.2.9. Trick Mode Support

Trick Modes are used by DASH clients in order to support fast forward, seek, rewind and other operations in which typically the media, especially video, is displayed in a speed other than the normal playout speed. In order to support such operations, it is recommended that the content author adds Representations at lower frame rates in order to support faster playout with the same decoding and rendering capabilities.

However, Representations targeted for trick modes are typically not be suitable for regular playout. If the content author wants to explicitly signal that a Representation is only suitable for trick mode cases, but not for regular playout, it the following is recommended:

- add an Adaptation Set that that only contains trick modes Representations
- annotate the Adaptation Set with an **EssentialProperty** descriptor or **SupplementalProperty** descriptor with URI "http://dashif.org/guidelines/trickmode" and the @value the value of @id attribute of the Adaptation Set to which these trick mode Representations belong. The trick mode Representations must be time-aligned with the Representations in the main Adaptation Set. The value may also be a white-space separated list of @id values. In this case the trick mode Adaptation Set is associated to all Adaptation Sets with the values of the @id.
- signal the playout capabilities with the attribute @maxPlayoutRate for each Representation in order to indicate the accelerated playout that is enabled by the signaled codec profile and level.

If an Adaptation Set in annotated with the **EssentialProperty** descriptor with URI "http://dashif.org/guidelines/trickmode then the DASH client shall not select any of the contained Representations for regular playout.

3.2.10. Adaptation Set Constraints

3.2.10.1. Introduction

Content in one Adaptation Set is constrained to enable and simplify switching across different Representations of the same source content. General Adaptation Set constraints allow sequencing of Media Segments from different Representations (“bitrate switching”) prior to a single audio or video decoder, typically requiring the video decoder to be reset to new decoding parameters at the switch point, such as a different encoded resolution or codec profile and level.

Bitstream Switching Adaptation Set constraints allow a switched sequence of Media Segments to be decoded without resetting the decoder at switch points because the resulting Segment stream is a valid track of the source type, so the decoder is not even aware of the switch. In order to signal that the Representations in an Adaptation Set are offered under these constraints, the attribute **AdaptationSet**@bitstreamSwitching may be set to true. In the following general requirements and recommendations are provided for content in an Adaptation Set in section 3.2.10.2 and specific requirements when the bitstream switching is set to true in section 3.2.10.3.

3.2.10.2. General

General Adaptation Set constraints require a client to process an Initialization Segment prior to the first Media Segment and prior to each Media Segment selected from a different Representation (a "bitrate switch").

Adaptation Sets shall contain Media Segments compatible with a single decoder that start with SAP type 1 or 2, and in time aligned Representations using the same @timescale, when multiple Representations are present.

Edit lists in Initialization Segments intended to synchronize the presentation time of audio and video should be identical for all Representations in an Adaptation Set.

Note: Re-initialization of decoders, decryptors, and display processors on some clients during bitrate switches may result in visible or audible artifacts. Other clients may evaluate the differences between Initialization Segments to minimize decoder reconfiguration and maintain seamless presentation equal to the encoded quality.

Additional recommendations and constraints may apply for encryption and media coding. For details, please check the relevant sections in this document, in particular section 6.2.5 and 7.5.4.

3.2.10.3. Bitstream Switching

A bitstream switching Adaptation Set is optimized for seamless decoding, and live streams that may change encoding parameters over time. A bitstream switching Adaptation Set may process an Initialization Segment one time from the highest bandwidth Representation in the Adaptation Set, and then process Media Segments from any other Representation in the same Adaptation Set without processing another Initialization Segment. The resulting sequence of an Initialization Segment followed by time sequenced Media Segments results in a valid ISO BMFF file with an elementary stream similar to a transport stream.

For all Representations within an Adaptation Set with `@bitstreamSwitching='true'`:

- the `Track_ID` shall be equal for all Representations
- Each movie fragment shall contain one track fragment

Note: Multiple Adaptation Sets may be included in an MPD that contain different subsets of the available Representations that are optimized for different decoder and screen limitations. A Representation may be present in more than one Adaptation set, for example a 720p Representation that is present in a 720p Adaptation Set may also be present in a 1080p Adaptation Set. The 720p Representation uses the same Initialization Segments in each Adaptation Set, but the 1080p Adaptation Set would require decoder and display configuration with the 1080p Initialization Segment.

Additional recommendation and constraints may apply for encryption and media coding. For details, please see below.

3.2.11. Media Time Information of Segment

The earliest presentation time may be estimated from the MPD using the segment availability start time minus the segment duration announced in the MPD.

The earliest presentation time may be accurately determined from the Segment itself.

If the Segment Index is present than this time is provided in the `earliest_presentation_time` field of the Segment Index. To determine the presentation time in the Period, the value of the attribute `@presentationTimeOffset` needs to be deducted.

If the Segment Index is not present, then the earliest presentation time is deduced from the ISO BMFF parameters, namely the movie fragment header and possibly in combination with the information in the Initialization Segment using the edit list.

The earliest presentation time in the Period for a Segment can be deduced from the decode time taking also into account the composition time offset, edit lists as well as presentation time offsets.

Specifically the following is the case to determine the earliest presentation time assuming that no edit list is present in the Initialization Segment:

- If the SAP type is 1, then the earliest presentation time is identical to the sum of the decode time and the composition offset of the first sample. The decode time of the first sample is determined by the base media decode time of the movie fragment.
- If the SAP type is 2, the first sample may not be the sample with the earliest presentation time. In order to determine the sample with the earliest presentation time, *this sample* is determined as the sample for which the sum of the decode time and the composition offset is the smallest within this Segment. Then the earliest presentation time of the Segment is the sum of the base media decode time and the sum of the decode time and the composition offset for *this sample*. Such an example is shown below.

In addition, if the presentation time needs to be adjusted at the beginning of a period, then the `@presentationTimeOffset` shall be used in order to set the presentation that is mapped to the start of the period. Content authoring shall be such that if edit lists are ignored, then the client can operate without timing and lip sync issues.

In the following examples, there is a sequence of I, P, and B frames, each with a decoding time delta of 10. The segmentation, presentation order and storage of the samples is shown in the table below. The samples are stored with the indicated values for their decoding time deltas and composition time offsets (the actual CT and DT are given for reference). The re-ordering occurs because the predicted P frames must be decoded before the bi-directionally predicted B frames. The value of DT for a sample is always the sum of the deltas of the preceding samples. Note that the total of the decoding deltas is the duration of the media in this track.

Example with closed GOP and SAP Type = 1:

Segment	/--	---	---	---	---	---	--\	/--	---	---	---	---	---	--\

<table>
<tr><td></td><td>I1</td><td>P4</td><td>B2</td><td>B3</td><td>P7</td><td>B5</td><td>B6</td><td>I8</td><td>P11</td><td>B9</td><td>B10</td><td>P14</td><td>B12</td><td>B13</td></tr>
<tr><td>Presentation Order</td><td colspan="14">|==| I1 B2 B3 P4 B5 B6 P7 |==| I8 B9 B10 P11 B12 B13 P14 |==|</td></tr>
<tr><td>Base media decode time</td><td colspan="7">0</td><td colspan="7">70</td></tr>
<tr><td>Decode delta</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td></tr>
<tr><td>DT</td><td>0</td><td>10</td><td>20</td><td>30</td><td>40</td><td>50</td><td>60</td><td>70</td><td>80</td><td>90</td><td>100</td><td>110</td><td>120</td><td>130</td></tr>
<tr><td>EPT</td><td colspan="7">10</td><td colspan="7">80</td></tr>
<tr><td>Composition time offset</td><td>10</td><td>30</td><td>0</td><td>0</td><td>30</td><td>0</td><td>0</td><td>10</td><td>30</td><td>0</td><td>0</td><td>30</td><td>0</td><td>0</td></tr>
<tr><td>CT</td><td>10</td><td>40</td><td>20</td><td>30</td><td>70</td><td>50</td><td>60</td><td>80</td><td>110</td><td>90</td><td>100</td><td>140</td><td>120</td><td>130</td></tr>
</table>

Example with closed GOP and SAP Type = 2:

<table>
<tr><td>Segment</td><td>/--</td><td>--</td><td>--</td><td>--</td><td>--</td><td>--\</td><td>/-</td><td>--</td><td>--</td><td>--</td><td>---</td><td>--\</td></tr>
<tr><td></td><td>I3</td><td>P1</td><td>P2</td><td>P6</td><td>B4</td><td>B5</td><td>I9</td><td>P7</td><td>P8</td><td>P12</td><td>B10</td><td>B11</td></tr>
<tr><td>Presentation Order</td><td colspan="12">|==| P1 P2 I3 B4 B5 P6 |==| P7 P8 I9 B10 B11 P12 |==|</td></tr>
<tr><td>Base media decode time</td><td colspan="6">0</td><td colspan="6">60</td></tr>
<tr><td>Decode Delta</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td></tr>
<tr><td>DT</td><td>0</td><td>10</td><td>20</td><td>30</td><td>40</td><td>50</td><td>60</td><td>70</td><td>80</td><td>90</td><td>100</td><td>110</td></tr>
<tr><td>EPT</td><td colspan="6">10</td><td colspan="6">70</td></tr>
<tr><td>Composition time offset</td><td>30</td><td>0</td><td>0</td><td>30</td><td>0</td><td>0</td><td>30</td><td>0</td><td>0</td><td>30</td><td>0</td><td>0</td></tr>
<tr><td>CT</td><td>30</td><td>10</td><td>20</td><td>60</td><td>40</td><td>50</td><td>90</td><td>70</td><td>80</td><td>120</td><td>100</td><td>110</td></tr>
</table>

Example with closed GOP and SAP Type = 2 and negative composition offset:

<table>
<tr><td>Segment</td><td>/--</td><td>--</td><td>--</td><td>--</td><td>--</td><td>--\</td><td>/-</td><td>--</td><td>--</td><td>--</td><td>---</td><td>--\</td></tr>
<tr><td></td><td>I3</td><td>P1</td><td>P2</td><td>P6</td><td>B4</td><td>B5</td><td>I9</td><td>P7</td><td>P8</td><td>P12</td><td>B10</td><td>B11</td></tr>
<tr><td>Presentation Order</td><td colspan="12">|==| P1 P2 I3 B4 B5 P6 |==| P7 P8 I9 B10 B11 P12 |==|</td></tr>
<tr><td>Base media decode time</td><td colspan="6">0</td><td colspan="6">60</td></tr>
<tr><td>Decode Delta</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td><td>10</td></tr>
<tr><td>DT</td><td>0</td><td>10</td><td>20</td><td>30</td><td>40</td><td>50</td><td>60</td><td>70</td><td>80</td><td>90</td><td>100</td><td>110</td></tr>
<tr><td>EPT</td><td colspan="6">0</td><td colspan="6">60</td></tr>
</table>

Composition offset	20	-10	-10	20	-10	-10	20	-10	-10	20	-10	-10
CT	20	0	10	50	30	40	80	60	70	110	90	100

For additional details refer to ISO/IEC 14496-12 [8] and ISO/IEC 23009-1 [1].

3.2.12. Content Offering with Periods

Content may be offered with a single Period. If content is offered with a single Period it is suitable to set `PSTART` to zero, i.e. the initialization segments get available at `START` on the server. However, other values for PSTART may be chosen.

Content with multiple Periods may be created for different reasons, for example:

- to enable splicing of content, for example for ad insertion,
- to remove or add certain Representations in an Adaptation Set,
- to remove or add certain Adaptation Sets,
- for robustness reasons as documented in detail in section 4.8.

For details on content offering with multiple Periods, please refer to the requirements and recommendations in section 4 and 5.

3.3. Client Implementation Requirements and Guidelines

3.3.1. Overview

The DASH-related aspects of the interoperability point as defined in section 3.2 can also be understood as permission for DASH clients that only implement the features required by the description to process the Media Presentation (MPD document and Segments). The detailed DASH-related client operations are not specified. Therefore, it is also unspecified how a DASH client exactly conforms. This document however provides guidelines on what is expected for conformance to this interoperability point. A minimum set of requirements is collected in section 3.3.4.

3.3.2. DASH Client Guidelines

The DASH-related aspects in DASH-IF IOPs as well as for the ISO BMFF based On-Demand and Live profiles of ISO/IEC 23009-1 are designed such that a client implementation can rely on relatively easy processes to provide an adaptive streaming service, namely:

- selection of the appropriate Adaptation Sets based on descriptors and other attributes
- initial selection of one Representation within each adaptation set
- download of (Sub)Segments at the appropriate time
- synchronization of different media components from different Adaptation Sets
- seamless switching of representations within one Adaptation Set

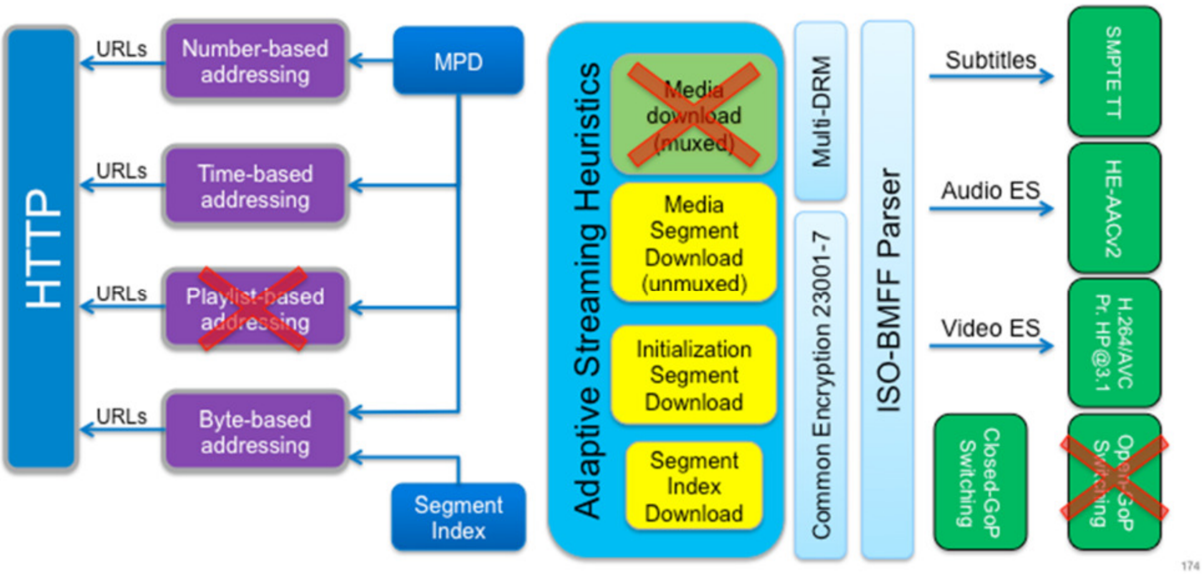


Figure 2 DASH aspects of a DASH-AVC/264 client compared to a client supporting the union of DASH ISO BMFF live and on-demand profile.

Figure 2 shows the DASH aspects of a DASH-AVC/264 client compared to a client supporting all features of the DASH ISO BMFF Live and On-Demand profile. The main supported features are:

- support of HTTP GET and partial GET requests to download Segments and Subsegments
- three different addressing schemes: number and time-based templating as well as byte range based requests.
- support of metadata as provided in the MPD and Segment Index
- download of Media Segments, Initialization Segments and Segment Index
- ISO BMFF parsing
- synchronized presentation of media components from different Adaptation Sets
- switching of video streams at closed GOP boundaries

3.3.3. Seamless switching

The formats defined in section 3.2 are designed for providing good user experience even in case the access bandwidth of the DASH Segment delivery or the cache varies. A key functionality is the ability that the DASH client can seamlessly switch across different Representations of the same media component. DASH clients should use the common timeline across different Representation representing the same media component to present one Representation up to a certain time t and continue presentation of another Representation from time *t* onwards. However, in practical implementations, this operation may be complex, as switching at time *t* may require parallel download and decoding of two Representations. Therefore, providing suitable switching opportunities in regular time intervals simplifies client implementations.

The formats defined in section 3.2 provide suitable switching opportunities at (sub)segment boundaries.

3.3.4. DASH Client Requirements

In order to ensure a minimum level of interoperability, a DASH-IF conforming client shall at least support the following features:

- The DASH client, if it switches, shall provide a seamless experience. A DASH shall be able to switch seamlessly at (sub)segment boundaries according to the definition in ISO/IEC 23009-1 [4], clause 4.5.1.
- If the scheme or the value for the following descriptor elements are not recognized and no equivalent other descriptor is present, the DASH client shall ignore the parent element:
    - **`FramePacking`**
    - **`Rating`**
    - **`EssentialDescriptor`**
    - **`ContentProtection`**

3.4. Transport and Protocol-Related Issues

3.4.1. General

Servers and clients operating in the context of the interoperability points defined in this document shall support the normative parts of HTTP/1.1 as defined in RFC2616 [22].

Specific requirements and recommendations are provided below.

3.4.2. Server Requirements and Guidelines

HTTP Servers serving segments should support suitable responses to byte range requests (partial GETs).

If an MPD is offered that contains Representations conforming to the ISO BMFF On-Demand profile, then the HTTP servers offering these Representations shall support suitable responses to byte range requests (partial GETs).

HTTP Servers may also support the syntax using Annex E of 23009-1 using the syntax of the second example in Annex E.3,

`BaseURL``@byteRange="$base$?$query$&range=$first$-$last$"`

3.4.3. Client Requirements and Guidelines

Clients shall support byte range requests, i.e. issue partial GETs to subsegments. Range requests may also be issued by using Annex E of 23009-1 using the syntax of the second example in Annex E.3,

`BaseURL``@byteRange="$base$?$query$&range=$first$-$last$"`

Clients shall follow the reaction to HTTP status and error codes as defined in section A.7 of ISO/IEC 23009-1.

Clients should support the normative aspects of the HTTP state management mechanisms (also known as Cookies) as defined in RFC 6265 [37] for first-party cookies.

3.5. Synchronization Considerations

In order to properly access MPDs and Segments that are available on DASH servers, DASH servers and clients should synchronize their clocks to a globally accurate time standard. Specifically it is expected that the Segment Availability Times as computed from the MPD according to ISO/IEC 23009-1 [5], section 5.3.9.5 and additional details in ISO/IEC 23009-3 [7], section 6.4 are accurately announced in the MPD.

Options to obtain timing for a DASH client are for example:

- Usage of NTP or SNTP as defined in RFC5905 [36].
- The Date general-header field in the HTTP header (see RFC2616 [22], section 14.18) represents the date and time at which the message was originated, and may be used as an indication of the actual time.

Anticipated inaccuracy of the timing source should be taken into account when requesting segments close to their segment availability time boundaries.

More details on advanced synchronization support is provided in section 4.7.

3.6. Considerations for Live Services

For interoperability aspects of live services, please refer to section 4.

3.7. Considerations on Ad Insertion

For interoperability aspects for ad insertion use cases, please refer to section 5.

4. Live Services

4.1. Introduction

MPEG-DASH [1] provides several tools to support live services. This section primarily provides requirements and recommendations for both, content authoring as well as client implementations.

For this purpose, this section

- clarifies and refines details of interoperability points when used with the features available in the 2012 edition of MPEG-DASH with respect to different service configurations and client implementations.
- defines one new interoperability point in order to address content authoring and client requirements to support a broad set of live services based on the features defined in the second edition (published 2014) of MPEG-DASH as well certain amendments thereof.

The main features and differences of these two modes are provided in the following table:

Feature	Simple	Main
Support of `MPD@type`	static, dynamic	static, dynamic
MPD updates	yes	yes
MPD updated triggered	by MPD attribute minimum update period	by Inband Event messages in the segments.
URL generation	based on MPD	based on MPD and segment information
Timeline gaps	based on MPD and for entire content	may be signalled individually for each Representation
Segments starts with	closed GOP	closed GOP
Support of Simple Live	Yes	No
Support of Main Live	Yes	Yes

To support the definition of the interoperability points, architectures and use cases were collected. These are documented in Annex X.

4.2. Overview Dynamic and Live Media Presentations

DASH Media Presentations with `MPD@type` set to `"dynamic"` enable that media is made available over time and its availability may also be removed over time. This has two major effects, namely

1. The content creator can announce a DASH Media Presentation for which not all content is yet available, but only gets available over time.
2. Clients are forced into a timed schedule for the playout, such that they follow the schedule as desired by the content author.

Dynamic services may be used for different types of services:

1. **Dynamic Distribution of Available Content**: Services, for which content is made available as dynamic content, but the content is entirely generated prior to distribution. In this case the details of the Media Presentation, especially the Segments (duration, URLs) are known and can be announced in a single MPD without MPD updates. This addresses use cases 2 and 3 in Annex B.
2. **MPD-controlled Live Service**: Services for which the content is typically generated on the fly, and the MPD needs to be updated occasionally to reflect changes in the service offerings. For such a service, the DASH client operates solely on information in the MPD. This addresses the use cases 4 and 5 in Annex B.
3. **MPD and Segment-controlled Live**: Services for which the content is typically generated on the fly, and the MPD may need to be updated on short notice to re-

flect changes in the service offerings. For such a service, the DASH client operates on information in the MPD and is expected to parse segments to extract relevant information for proper operation. This addresses the use cases 4 and 5, but also takes into account the advanced use cases.

Dynamic and Live services are typically controlled by different client transactions and server-side signaling.

For initial access to the service and joining the service, an MPD is required. MPDs may be accessed at join time or may have been provided earlier, for example along with an Electronic Service Guide. The initial MPD or join MPD is accessed and processed by the client and the client having an accurate clock that is synchronized with the server can analyze the MPD and extract suitable information in order to initiate the service. This includes, but is not limited to:

- identifying the currently active Periods in the service and the Period that expresses the live edge (for more details see below)
- selecting the suitable media components by selecting one or multiple Adaptation Sets. Within each Adaptation Set selecting an appropriate Representation and identifying the live edge segment in each Representations. The client then issues requests for the Segments.

The MPD may be updated on the server based on certain rules and clients consuming the service are expected to update MPDs based on certain triggers. The triggers may be provided by the MPD itself or by information included in Segments. Depending on the service offering, different client operations are required as shown in Figure 3.

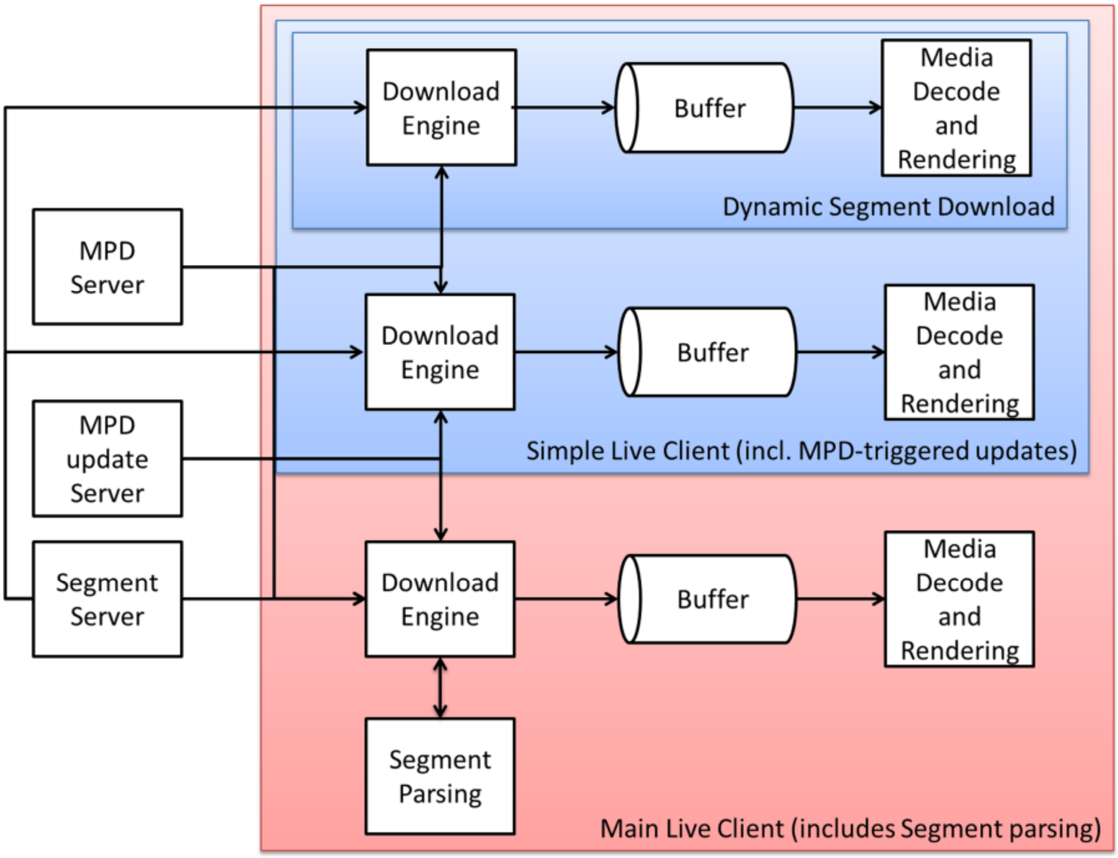


Figure 3 Different Client Models

The basic functions a live clients describes in this document are as follows:

1. **Dynamic Segment Download:** This function creates a list of available Segments based on a single MPD and joins the service by downloading Segments at the live edge or may use the Segments that are available in the time shift buffer.

2. **Simple Live Client**: This client includes the dynamic segment download function and enables updates of the MPD based on information in the MPD in order to extend the Segment list at the live edge. MPDs are refetched and revalidated when the currently available MPD expires, i.e. an expired MPD can no longer be used for Segment URL generation.

3. **Main Live Client**: This client includes all features of the simple Live DASH client. In addition it generates Segment URLs and it updates the MPD based on information in the Segments if the service offering provides this feature. MPDs are

refetched and revalidated when the currently available MPD expires based on expiry information in the Segments.

Requirements and recommendations for the dynamic segment download functions are defined in in section 4.3.

Requirements and recommendations for simple live service offerings and corresponding clients are defined in section 4.4.

Requirements and recommendations for main live service offerings and corresponding clients are defined in section 4.5.

Requirements and recommendations when offering live services as on-demand are provided in section 4.6.

Requirements and recommendations for client-server timing synchronization are defined in section 4.7.

Requirements and recommendations for robust service offerings and corresponding clients are defined in section 4.8.

Interoperability Points are defined in section 4.9.

4.3. Dynamic Segment Download

4.3.1. Background and Assumptions

The dynamic segment download function is a key component of live services, In addition, the dynamic segment download function may also be used for scheduling a playout. In the remainder of this subsection, it is assumed that the client has access to a single instance of an MPD and all information of the entire Media Presentation is contained in the MPD.

We refer to this service as dynamic service as the main feature is that the Segments are made available over time following the schedule of the media timeline.

Dynamic services are primarily documented in order to provide insight into the timing model of Segment availabilities. This forms the basis for live services and explains the key concepts and rules for Segment availabilities.

4.3.2. Preliminaries

4.3.2.1. MPD Information

If the Media Presentation is of type dynamic, then Segments have different Segment availability times, i.e. the earliest time for which the service provider permits the DASH client to issue a request to the Segment and guarantees, under regular operation modes, that the client gets a 200 OK response for the Segment. The Segment availability times for each Representation can be computed based on the information in an MPD.

For a dynamic service the MPD should at least contain information as available in Table 3. Information included there may be used to compute a list of announced Segments, Segment Availability Times and URLs.

Assume that an MPD is available to the DASH client at a specific wall-clock time *NOW*. It is assumed that the client and the DASH server providing the Segments are synchronized to wall-clock, either through external means or through a specific client-server synchronization. Details on synchronization are discussed in section 4.7.

Assuming synchronization, the information in the MPD can then be used by the client at time *NOW* to derive the availability (or non-availability) of Segments on the server.

Table 3 -- Information related to Segment Information and Availability Times for a dynamic service

MPD Information	Status	Comment
MPD@type	mandatory, set to "dynamic"	the type of the Media Presentation is dynamic, i.e. Segments get available over time.
MPD@availabilityStartTime	mandatory	the start time is the anchor for the MPD in wall-clock time. The value is denoted as *AST* in the following.
MPD@mediaPresentationDuration	mandatory (for the considered use cases)	provides the duration of the Media Presentation.
MPD@suggestedPresentationDelay	optional, but recommended	suggested presentation delay as delta to segment availability start time. The value is denoted as *SPD*. Details on the setting and usage of the parameter is provided in the following.
MPD@minBufferTime	mandatory	minimum buffer time, used in conjunction with the @bandwidth attribute of each Representation. The value is denoted as *MBT*. Details on the setting and usage of the parameter is provided in the following.
MPD@timeShiftBufferDepth	optional, but recommended	time shift buffer depth of the media presentation. The value is denoted as *TSB*. Details on the setting and usage of the parameter is provided in the following.
Period@start	Mandatory for the first Period in the MPD	the start time of the Period relative to the MPD availability start time.

SegmentTemplate@media	mandatory	The template for the Media Segment assigned to a Representation.
SegmentTemplate@startNumber	optional default	number of the first segment in the Period assigned to a Representation
SegmentTemplate@timescale	optional default	timescale for this Representation.
SegmentTemplate@duration	exactly one of **SegmentTemplate**@duration or **SegmentTemplate.SegmentTimeline** must be present per Representation.	the duration of each Segment in units of a time.
SegmentTemplate.SegmentTimeline		

4.3.2.2. Segment Information Derivation

4.3.2.2.1. Introduction

Based on an MPD including information as documented in Table 3 and available at time *NOW* on the server, a synchronized DASH client derives the information of the list of Segments for each Representation in each Period. This section only describes the information that is expressed by the values in the MPD. The generation of the information on the server and the usage of the information in the client is discussed in section 4.3.3 and 4.3.4, respectively.

MPD information is provided in subsection 4.3.2.2.3. The Period based information is documented in sub-section 4.3.2.2.4, and the Representation information is documented in sub-section 4.3.2.2.5.

4.3.2.2.2. Definitions

The following definitions are relevant and aligned with ISO/IEC 23009-1:

- available Segment is a Segment that is accessible at its assigned HTTP-URL. This means that a request with an HTTP GET to the URL of the Segment results in a reply of the Segment and 2xx status code.
- valid Segment URL is an HTTP-URL that is promised to reference a Segment during its Segment availability period.
- *NOW* is a time that is expressing the time on the content server as wall-clock time. All information in the MPD related to wall-clock is expressed as a reference to the time *NOW*.

4.3.2.2.3. MPD Information

For a dynamic service without MPD updates, the following information shall be present and not present in the MPD (also please refer to Table 3):

- The **MPD**@type shall be set to "dynamic".
- The **MPD**@mediaPresentationDuration shall be present, or the **Period**@duration of the last Period shall be present.

- The **MPD**@minimumUpdatePeriod shall not be present.

Furthermore, it is recommended to provide a value for **MPD**@timeShiftBufferDepth and **MPD**@suggestedPresentationDelay.

4.3.2.2.4. Period Information

Each Period is documented by a **Period** element in the MPD. An MPD may contain one or more Periods. In order to document the use of multiple Periods, the sequence of Period elements is expressed by an index *i* with *i* increasing by 1 for each new Period element.

Each regular Period *i* in the MPD is assigned a

- Period start time *PSwc*[*i*] in wall-clock time,
- Period end time *PEwc*[*i*], in wall-clock time.

Note: An MPD update may extend the Period end time of the last Period. For details refer to section 4.4.

The Period start time *PSwc*[*i*] for a regular Period *i* is determined according to section 5.3.2.1 of ISO/IEC 23009-1:

- If the attribute @start is present in the **Period**, then *PSwc*[*i*] is the sum of *AST* and the value of this attribute.
- If the @start attribute is absent, but the previous **Period** element contains a @duration attribute then the start time of the Period is the sum of the start time of the previous Period *PSwc*[*i*] and the value of the attribute @duration of the previous Period. Note that if both are present, then the @start of the new Period takes precedence over the information derived from the @duration attribute.

The Period end time *PEwc*[*i*] for a regular Period *i* is determined as follows:

- If the Period is the last one in the MPD, the time *PEwc*[*i*] is obtained as
    - the sum of *AST* and Media Presentation Duration *MPDur*, with *MPDur* the value of **MPD**@mediaPresentationDuration if present, or the sum of *PSwc*[*i*] of the last Period and the value of **Period**@duration of the last Period.
- else
    - the time *PEwc*[*i*] is obtained as the Period start time of the next Period, i.e. *PEwc*[*i*] = *PSwc*[*i*+1].

4.3.2.2.5. Representation Information

Based on such an MPD at a specific time *NOW*, a list of Segments contained in a Representation in a Period *i* with Period start time *PSwc*[*i*] and Period end time *PEwc*[*i*] can be computed.

If the **SegmentTemplate.SegmentTimeline** is present and the **SegmentTemplate**@duration is not present, the **SegmentTimeline** element contains N_S **S** elements indexed with s=1, ..., N_S, then let

- *ts* the value of the @timescale attribute
- *t*[*s*] be the value of @t of the *s*-th **S** element,

- *d*[*s*] be the value of `@d` of the *s*-th **S** element
- *r*[*s*] be,
  - if the `@r` value is greater than or equal to zero
    - one more than the value of `@r` of the *s*-th **S** element. Note that if `@r` is smaller than the end of this segment timeline element, then this Representation contains gaps and no media is present for this gap.
  - else
    - if *t*[*s*+1] is present, then *r*[*s*] is the ceil of (*t*[*s*+1] - *t*[*s*])/*d*[*s*]
    - else *r*[*s*] is the ceil of (*PEwc*[*i*] - *PSwc*[*i*] - *t*[*s*]/*ts*)**ts*/*d*[*s*])

If the **SegmentTemplate**`@duration` is present and the **SegmentTemplate.SegmentTimeline** is not present, then

- N_S=1,
- *ts* the value of the `@timescale` attribute
- *t*[*s*] is 0,
- the *d*[*s*] is the value of `@duration` attribute
- *r*[*s*] is the ceil of (*PEwc*[*i*] - *PSwc*[*i*] - *t*[*s*]/*ts*)**ts*/*d*[*s*])

4.3.2.2.6. Media Time Information of Segment

Each Media Segment at position *k=1,2, ...* for each Representation has assigned an earliest media presentation time *EPT*[*k,r,i*] and an accurate segment duration *SDUR[k,r,j]*, all measured in media presentation time.
The earliest presentation time may be estimated from the MPD using the segment availability start time minus the segment duration announced in the MPD.
The earliest presentation time may be accurately determined from the Segment itself.

For details on the derivation of the earliest presentation time, see section 3.2.11.

4.3.2.2.7. Segment List Parameters

For each Period *i* with Period start time *PSwc*[*i*] and Period end time *PEwc*[*i*] and each Representation *r* in the Period the following information can be computed:

- the presentation time offset described in the MPD, o[*i,r*]
- the number of the first segment described in the MPD, k1[*i,r*]
- the number of the last segment described in the MPD, k2[*i,r*]
- segment availability start time of the initialization segment SAST[*0,i,r*]
- segment availability end time of the initialization segment SAET[*0,i,r*]
- segment availability start time of each media segment SAST[*k,i,r*], *k*=k1, ..., k2
- segment availability end time of each media segment SAET[*k,i,r*], *k*=k1, ..., k2
- segment duration of each media segment SD[*k,i,r*], k=k1, ..., k2
- the URL of each of the segments, URL[*k,i,r*]

In addition,

- the latest available Period *i*[*NOW*] and the latest segment available at the server k[*NOW*] can be computed.
- the earliest available Period *i**[*NOW*] and the earliest segment available at the server k*[*NOW*] can be computed.

Based on the above information, for each Representation *r* in a Period *i*, the segment availability start time *SAST*[*k,i,r*], the segment availability end time of each segment *SAET*[*k,i,r*], the segment duration of each segment *SD*[*k,i,r*], and the URL of each of the segments, *URL*[*k,i,r*] within one Period *i* be derived as follows using the URL Template function `URLTemplate`(`ReplacementString`, Address) as documented in subsection 4.3.2.2.8:

- *k*=0
- *SAST*[*0,i,r*] = *PSwc*[*i*]
- for *s=1, ...* N_s [*i,r*]
    - *k* = *k* + 1
    - *SAST*[*k,i,r*] = *PSwc*[*i*] + (*t*[*s,i,r*] + *d*[*s,i,r*] - o[*i,r*])/*ts*
    - *SD*[*k,i,r*] = *d*[*s,i,r*]/*ts*
    - *SAET*[*k,i,r*] = SAST[*k,i,r*] + *TSB* + *d*[*s,i,r*]/*ts*
    - if **`SegmentTemplate`**`@media` contains `$Number$`
        - *Address*=`@startNumber`
        - *URL*[*k,i,r*] = `URLTemplate` (`$Number$`, *Address*)

      else
        - *Address* = *t*[*s,i,r*]
        - *URL*[*k,i,r*] = `URLTemplate` (`$Time$`, *Address*)
    - for *j* = 1, ..., *r*[*s,i,r*]
        - *k* = *k* + 1
        - *SAST*[*k,i,r*] = SAST[*k-1,i,r*] + *d*[*s,i,r*]/*ts*
        - *SAET*[*k,i,r*] = SAST[*k,i,r*] + *TSB* + *d*[*s,i,r*] /*ts*
        - *SD*[*k,i,r*] = *d*[*s,i,r*] /*ts*
        - if **`SegmentTemplate`**`@media` contains `$Number$`
            - *Address* = *Address* + 1
            - *URL*[*k,i,r*] = `URLTemplate` (`$Number$`, *Address*)

          else
            - Address = Address + *d*[*s,i,r*]
            - *URL*[*k,i,r*] = `URLTemplate` (`$Time$`, *Address*)
- k2[*i,r*] = *k*
- *SAET*[*0,i,r*] = *SAET*[k2[*i,r*],*i,r*]

Note that not all segments documented above may necessarily be accessible at time *NOW*, but only those that are within the segment availability time window.

Hence, the number of the first media segment described in the MPD for this Period, k1[*i,r*], is the smallest *k*=1, 2, ... for which *SAST*[*k,i,r*] >= *NOW*.

The latest available Period *i*[*NOW*] is the Period *i* with the largest *PEwc*[*i*] and *PEwc*[*i*] is smaller than or equal to *NOW*.

The latest available segment *k*[*NOW*] available for a Representation of Period *i*[*NOW*] is the segment with the largest *k*=0,1,2,... such that *SAST*[*k,i,r*] is smaller than or equal to *NOW*. Note that this contains the Initialization Segment with *k*=0 as not necessarily any media segment may yet be available for Period *i*[*NOW*]. In this case, last media segment k2[*i*[*NOW*]-1,*r*], i.e., the last media segment of the previous Period is the latest accessible media Segment.

4.3.2.2.8. URL Generation with Segment Template

The function URL Template function `URLTemplate`(`ReplacementString`, Address) generates a URL. For details refer to ISO/IEC 23009-1 [1], section 5.3.9.4. Once the Segment is generated, processing of the Base URLs that apply on this segment level is done as defined in ISO/IEC 23009-1, section 5.6.

4.3.2.2.9. Synchronized Playout and Seamless Switching

In order to achieve synchronized playout across different Representations, typically from different Adaptation Sets, the different Representations are synchronized according to the presentation time in the Period. Specifically, the earliest presentation time of each Segment according to section 4.3.2.2.6 determines the playout of the Segment in the Period and therefore enables synchronized playout of different media components as well as seamless switching within one media component.

4.3.3. Service Offering Requirements and Guidelines

4.3.3.1. General Service Offering Requirements

For dynamic service offerings, the MPD shall conform to DASH-IF IOP as defined in section 3 and shall at least contain the mandatory information as documented in Table 3.

If such an MPD is accessible at time *NOW* at the location **`MPD.Location`**, then

- all Segments for all Representations in all Periods as announced in an MPD shall be available latest at the announced segment availability start time *SAST*[*k,i,r*] at all *URL*[*k,i,r*] as derived in section 4.3.2.2;
- all Segments for all Representations in all Periods as announced in an MPD shall at least be available until the announced segment availability end time *SAET*[*k,i,r*] at all *URL*[*k,i,r*] as derived in section 4.3.2.2;
- for all Media Segments for all Representations in all Periods as announced in an MPD the Segment in this Period is available prior to the sum of Period start, earliest presentation time and segment duration, i.e. *SAST*[*k,i,r*] <= *PSwc*[*i*] + *SD*[*k,r,i*] + *EPT*[*k,r,i*];
- if a Media Segments with segment number *k* is delivered over a constant bitrate channel with bitrate equal to value of the `@bandwidth` attribute then each presentation time *PT* is available at the client latest at time with a delay of at most *PT* + *MBT*.

4.3.3.2. Dynamic Service Offering Guidelines

4.3.3.2.1. Introduction

In order to offer a simple dynamic service for which the following details are known in advance,

- start at wall-clock time `START`,
- exact duration of media presentation `PDURATION`,
- location of the segments for each Representation at " `http://example.com/$RepresentationID$/$Number$`",

a service provide may offer an MPD as follows:

Table 4 – Basic Service Offering

MPD Information	Value
MPD@type	dynamic
MPD@availabilityStartTime	START
MPD@mediaPresentationDuration	PDURATION
MPD@suggestedPresentationDelay	SPD
MPD@minBufferTime	MBT
MPD@timeShiftBufferDepth	TSB
MPD.BaseURL	"http://example.com/"
Period@start	PSTART
Representation@bandwidth	BW
SegmentTemplate@media	"$RepresentationID$/$Number$"
SegmentTemplate@startNumber	1
SegmentTemplate@duration	SDURATION

Note that the setting of capitalized parameters is discussed in section 4.3.3.2.2.

According to the work-flow shown in Annex B:

- the MPD is generated and published prior to time `START` such that DASH clients may access it prior to the start of the Media Presentation.
- no redundant tools are considered.
- the encoder and the segmenter generate segments of duration `SDURATION` and publish those on the origin server, such that they are available at URL[*k*] latest at their announced segment availability start time SAST[*k*].

Based on the details in section 4.3.2.2, the Segment Information is derived as:

- k1 = 1
- k2 = `ceil(PDURATION/SDURATION)`
- for k = 1, ..., k2
  - SAST[*k*] = `START` + `PSTART` + *k**`SDURATION`
  - SAET[*k*] = SAST[*k*] + `TSB` + `SDURATION`

- SD[*k*] = `SDURATION`
- URL[*k*] = `http://example.com/$RepresentationID$/k`

- The segment availability times of the Initialization Segment are as follows:
  - SAST[*0*] = `START` + `PSTART`
  - SAET[0] = SAET[k2]

4.3.3.2.2. Basic Parameter Settings

In the following recommendations are provided for the

- Time Shift Buffer Depth (`TSB`):
  - If the content should be consumed at the live edge, then the time shift buffer depth should be set short. However, the `TSB` should not be smaller than the recommended value of 4*`SDURATION` and 6 seconds in media time in order for the client to do some prebuffering in more difficult network conditions.
  - If no restrictions on the accessibility of the content are provided, then the `TSB` may be set to a large value that even exceeds `PDURATION`.
- Suggested Presentation Delay (`SPD`)
  - If synchronized play-out with other devices adhering to the same rule is desired and/or the service provider wants to define the typical live edge of the program, then this value should be provided. The service provider should set the value taking into account at least the following:
    - the desired end-to-end latency
    - the typical required buffering in the client, for example based on the network condition
    - the segment duration `SDURATION`
    - the time shift buffer depth `TSB`
  - A reasonable value may be 2 to 4 times of the segment duration `SDURATION`, but the time should not be smaller than 4 seconds in order for the client to maintain some buffering.
- Segment Duration (`SDURATION`)
  - The segment duration typically influences the end-to-end latency, but also the switching and random access granularity as in DASH-264/AVC each segment starts with a stream access point which can also be used as s switch point. The service provider should set the value taking into account at least the following:
    - the desired end-to-end latency
    - the desired compression efficiency
    - the start-up latency
    - the desired switching granularity
    - the desired amount of HTTP requests per second
    - the variability of the expected network conditions
  - Reasonable values for segment durations are between 1 second and 10 seconds.
- Minimum Buffer Time (`MBT`) and bandwidth (`BW`)

- the value of the minimum buffer time **does not provide any instructions to the client on how long to buffer the media**. This aspect is covered in 4.3.4.4. The value describes how much buffer a client should have under ***ideal*** network conditions. As such, `MBT` is not describing the burstiness or jitter in the network, it is describing the burstiness or jitter in the **content encoding**. Together with the `BW` value, it is a property of the content. Using the "leaky bucket" model, it is the size of the bucket that makes `BW` true, given the way the content is encoded.
- The minimum buffer time provides information that for each Stream Access Point (and in the case of DASH-IF therefore each start of the Media Segment), the property of the stream: If the Representation (starting at any segment) is delivered over a constant bitrate channel with bitrate equal to value of the `BW` attribute then each presentation time *PT* is available at the client latest at time with a delay of at most *PT* + *MBT*.
- In the absence of any other guidance, **the** `MBT` **should be set** to the maximum GOP size (coded video sequence) of the content, which quite often is identical **to the maximum segment duration**. The *MBT* may be set to a smaller value than maximum segment duration, but should not be set to a higher value.

In a simple and straightforward implementation, a DASH client decides downloading the next segment based on the following status information:

- the currently available buffer in the media pipeline, *buffer*
- the currently estimated download rate, *rate*
- the value of the attribute `@minBufferTime`, *MBT*
- the set of values of the `@bandwidth` attribute for each Representation i, *BW*[*i*]

The task of the client is to select a suitable Representation *i*.

The relevant issue is that starting from a SAP on, the DASH client can continue to playout the data. This means that at the current time it does have *buffer* data in the buffer. Based on this model the client can download a Representation *i* for which *BW*[*i*] ≤ *rate*buffer/MBT* without emptying the buffer.

Note that in this model, some idealizations typically do not hold in practice, such as constant bitrate channel, progressive download and playout of Segments, no blocking and congestion of other HTTP requests, etc. Therefore, a DASH client should use these values with care to compensate such practical circumstances; especially variations in download speed, latency, jitter, scheduling of requests of media components, as well as to address other practical circumstances.

One example is if the DASH client operates on Segment granularity. As in this case, not only parts of the Segment (i.e., `MBT`) needs to be downloaded, but the entire Segment, and if the `MBT` is smaller than the Segment duration, then rather the segment duration needs to be used instead of the `MBT` for the required buffer size and the download scheduling, i.e. download a Representation *i* for which *BW*[*i*] ≤ *rate*buffer/max_segment_duration*.

For low latency cases, the above parameters may be different.

4.3.3.2.3. Example

Assume a simple example according to Table 7.

Table 5 – Basic Service Offering

MPD Information	Value
MPD@type	dynamic
MPD@availabilityStartTime	START
MPD@mediaPresentationDuration	43sec
MPD@suggestedPresentationDelay	15sec
MPD@minBufferTime	5sec
MPD@timeShiftBufferDepth	25sec
MPD.BaseURL	"http://example.com/"
Period@start	0
SegmentTemplate@media	"$RepresentationID$/$Num-ber$"
SegmentTemplate@startNumber	1
SegmentTemplate@duration	5sec

Based on the derivation in section 4.3.3.2.1, the following holds:

- k1 = 1, k2 = 9
- for k = 1, ..., k2
  - SAST[*k*] = START + *k**5sec
  - SAET[*k*] = SAST[*k*] + 30sec
  - URL[*k*] = http://example.com/1/k
- The segment availability times of the Initialization Segment are as follows:
  - SAST[*0*] = START
  - SAET[0] = START + 75 sec

Figure 4 shows the availability of segments on the *server* for different times NOW. In particular, before START no segment is available, but the segment URLs are valid. With time *NOW* advancing, segments get available.

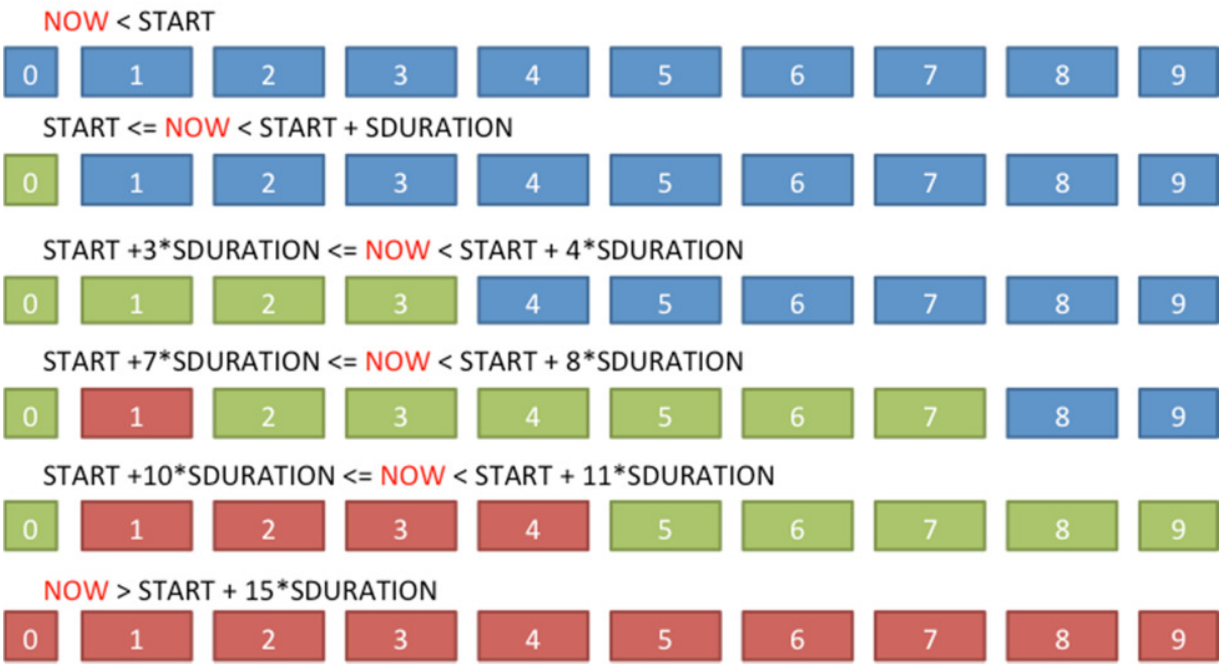


Figure 4 Segment Availability on the Server for different time NOW (blue = valid but not yet available segment, green = available Segment, red = unavailable Segment)

4.3.3.3. Content Offering with Periods

4.3.3.3.1. General

For content offered within a Period, and especially when offered in multiple Periods, then the content provider should offer the content such that actual media presentation time is as close as possible to the actual Period duration. It is recommended that the Period duration is the maximum of the presentation duration of all Representations contained in the Period.

A typical Multi-Period Offering is shown in Table 6. This may for example represent a service offering where main content provided in Period 1 and Period 3 are interrupted by an inserted Period 2.

Table 6 Multi-Period Service Offering

MPD Information	Value
MPD@type	dynamic
MPD@availabilityStartTime	START
MPD@mediaPresentationDuration	PDURATION
MPD@suggestedPresentationDelay	SPD
MPD@minBufferTime	MBT
MPD@timeShiftBufferDepth	TSB
MPD.BaseURL	"http://example.com/"
Period@start	PSTART
SegmentTemplate@media	"1/$RepresentationID$/$Number$"

SegmentTemplate@startNumber	1
SegmentTemplate@duration	SDURATION1
Period@start	PSTART2
SegmentTemplate@media	"2/$RepresentationID$/$Number$"
SegmentTemplate@startNumber	1
SegmentTemplate@duration	SDURATION2
Period@start	PSTART3
SegmentTemplate@media	"1/$RepresentationID$/$Number$"
SegmentTemplate@startNumber	STARTNUMBER2
SegmentTemplate@duration	SDURATION1
SegmentTemplate@presentationTimeOffset	PTO

The work flow for such a service offering is expected to be similar to the one in section 4.3.2.2.1.

Based on the details in section 4.3.2.2, the Segment Information is derived as:

- Period 1
  - PSwc[1] = `START` + `PSTART`
  - PEwc[1] = `START` + `PSTART2`
  - k1 = 1
  - k2 = `ceil((PSTART2-PSTART1)/SDURATION)`
  - for k = 1, ..., k2
    - SAST[*k*] = PSwc[1] + *k**`SDURATION`
    - SAET[*k*] = SAST[*k*] + `TSB` + `SDURATION`
    - SD[*k*] = `SDURATION`
    - URL[*k*] = `http://example.com/1/$RepresentationID$/k`
  - SAST[0] = PSwc[1]
  - SAET[0] = SAET[k2]
- Period 2
  - PSwc[2] = `START` + `PSTART2`
  - PEwc[2] = `START` + `PSTART3`
  - k1 = 1
  - k2 = `ceil((PSTART3-PSTART2)/SDURATION2)`
  - for k = 1, ..., k2
    - SAST[*k*] = PSwc[2] + *k**`SDURATION2`
    - SAET[*k*] = SAST[*k*] + `TSB` + `SDURATION2`
    - SD[*k*] = `SDURATION2`
    - URL[*k*] = `http://example.com/2/$RepresentationID$/k`
  - SAST[0] = PSwc[2]
  - SAET[0] = SAET[k2]
- Period 3
  - PSwc[3] = `START` + `PSTART3`
  - PEwc[3] = `START` + `PDURATION`
  - k1 = 1

    - k2 = `ceil((PSTART3-PDURATION)/SDURATION1)`
    - for k = 1, ..., k2
        - SAST[*k*] = PSwc[3] + *k**`SDURATION1`
        - SAET[*k*] = SAST[*k*] + `TSB` + `SDURATION1`
        - SD[*k*] = `SDURATION1`
        - URL[*k*] = "`http://example.com/1/$RepresentationID$/(k+STARTNUMBER2-1)`"
    - SAST[0] = PSwc[3]
    - SAET[0] = SAET[k2]

Note that the number *k* describes position in the Period. The actual number used in the segment template increased by the one less than the actual start number.

4.3.3.3.2. Continuous Period Offering

> Note: This is aligned with Amd.3 of ISO/IEC 23009-1:2014 [5] and may be referenced in a future version of this document.

In certain circumstances the Content Provider offers content in the next Period that is a continuation of the content in the previous Period, possibly in the immediately following Period or in a later Period. The latter case applies for example after an advertisement Period had been inserted. The content provider may express that the media components contained in two Adaptation Sets in two different Periods are *associated* by assigning equivalent Asset Identifiers to both Periods and by identifying both Adaptation Sets with identical value for the attribute `@id`.

If Adaptation Sets in two different Periods are *associated*, then the Adaptation Set parameters defined in ISO/IEC 23009-1, section 5.3.3.1, must be identical for the two Adaptation Sets.

Furthermore, two Adaptation Sets in one MPD are *period-continuous* if all of the following holds:

- The Adaptation Sets are associated.
- The `@presentationTimeOffset` is present or can be inferred as 0 for all Representations in both Adaptation Sets.
- Within one Adaptation Set, the value of `@presentationTimeOffset` is identical for all Representations.
- The sum of the value of the `@presentationTimeOffset` and the *presentation duration* of all Representations in one Adaptation Set are identical to the value of the `@presentationTimeOffset` of the other Adaptation Set.
- If Representations in both Adaptation Sets have the same value for `@id`, then they shall have functionally equivalent Initialization Segments, i.e. the Initialization Segment may be used to continue the play-out the Representation.

Content authors should signal *period-continuous* Adaptation Sets by signalling the presentation duration. The *presentation duration* of a Representation is the difference between the end presentation time of the Representation and the earliest presentation time of the Representation. The presentation time duration has the same unit as presentation time offset, i.e. @timescale, and expresses the exact presentation duration of the Representation.

The presentation duration may be signalled by

- A supplemental descriptor with @scheme_id_URI set to "urn:mpeg:dash:period_continuity:2014" may be provided for an Adaptation Set with
    - the @value of the descriptor, PID, matching the value of an @id of a Period that is contained in the MPD,
    - the value of the **AdaptationSet**@id being AID,
    - the value of the @presentationTimeOffset for this Adaptation Set is provided and is PTO.

If this signal is present, then for the Period with the value of the **Period**@id being PID and for the Adaptation Set with **AdaptationSet**@id being AID, the presentation duration of each Representation in this Adaptation Set is obtained as the difference of PTO minus the value of the @presentationTimeOffset.

Content authors should offer an MPD with *period-continuous* Adaptation Sets if the MPD contains Periods with identical Asset Identifiers.

4.3.3.4. Content Offering with Segment Timeline

4.3.3.4.1. Basic Operation

In order to offer a dynamic service that takes into account

- variable segment durations
- gaps in the segment timeline of one Representation,

the Segment timeline as defined in ISO/IEC 23009-1, section 5.3.9.6 may be used as an alternative to the @duration attribute as shown in section 4.3.3.2.

Table 7 – Service Offering with Segment Timeline

MPD Information	Value
MPD@type	dynamic
MPD@availabilityStartTime	START
MPD@mediaPresentationDuration	PDURATION
MPD@suggestedPresentationDelay	SPD
MPD@minBufferTime	MBT
MPD@timeShiftBufferDepth	TSB
MPD.BaseURL	"http://example.com/"
Period@start	PSTART
SegmentTemplate@media	"$RepresentationID$/$Number$"
SegmentTemplate@startNumber	1
SegmentTemplate.SegmentTimeline	t[i], n[i], d[i], r[i]

According to the work-flow shown in Annex B:

- the MPD is generated and published prior to time `START` such that DASH clients may access it prior to the start of the Media Presentation.
- no redundant tools are considered.
- the encoder and the segmenter generally should generate segments of constant duration `SDURATION` and publish those on the origin server, such that they are available at URL[*k*] latest at their announced segment availability start time SAST[*k*]. However, the server may offer occasional shorter segments for encoding optimizations, e.g. at scene changes, or segment gaps (for details see section 6). If such an irregular segment is published the MPD needs to document this by a new **S** element in the segment timeline.

If the segment timeline is used and the `$Time$` template is used, then the times in the MPD shall accurately present media internal presentation times.

If the segment timeline is and the `$Number$` template is used, then the MPD times shall at most deviate from the earliest presentation time documented in the MPD by 0.5sec.

Based on these considerations, it is not feasible to operate with a single MPD if the content is not yet known in advance. However, pre-prepared content based on the segment timeline may be offered in a dynamic fashion. The use of the Segment Timeline is most suitable for the case where the MPD can be updated. For details refer to section 4.4.

4.3.3.4.2. Basic Parameter Settings

The parameters for `TSB` and `SPD` should be set according to section 4.3.3.2.2. The segment duration `SDURATION` may be set according to section 4.3.3.2.2, but it should be considered that the service provider can offer shorter segments occasionally.

4.3.3.5. Joining Recommendation

By default, an MPD with **MPD**`@type="dynamic"` suggests that the client would want to join the stream at the live edge, therefore to download the latest available segment (or close to, depending on the buffering model), and then start playing from that segment onwards.

However there are circumstances where a dynamic MPD might be used with content intended for playback from the start, or from another position. For example, when a content provider offers ‘start again’ functionality for a live program, the intention is to make the content available as an on-demand program, but not all the segments will be available immediately.

This may be signalled to the DASH client by including an MPD Anchor, with either

- the `t` parameter, or
- both the `period` and `t` parameter, in the MPD URL provided to the DASH client, or
- the `utc` parameter, for details refer to Amd.3 of ISO/IEC 23009-1:2014 [5].

The format and behaviour of MPD Anchors is defined in section C.4 of ISO/IEC 23009-1. Specifically the `utc` parameter is defined in Amd.3 of ISO/IEC 23009-1:2014 [5].

For example to start from the beginning of the MPD the following would be added to the end of the MPD URL provided to the DASH client:

```
#t=0
```

Or to start from somewhere other than the start, in this case 50 minutes from the beginning of the period with Period ID "`program_part_2`":

```
#period=program_part_2&t=50:00
```

Notes:

- as per section C.4 of ISO/IEC 23009-1 the time indicated using the t parameter is as per the field definition of the W3C Media Fragments Recommendation v1.0 section 4.2.1.
- the period ID has to be URL encoded/decoded as necessary and needs to match one of the **`Period`**`@id` fields in the MPD.

Where an MPD Anchor is used it should refer to a time for which segments are currently available in the MPD.

4.3.4. Client Operation, Requirements and Guidelines

4.3.4.1. Basic Operation for Single Period

A DASH client is guided by the information provided in the MPD. A simple client model is shown in Figure 5.

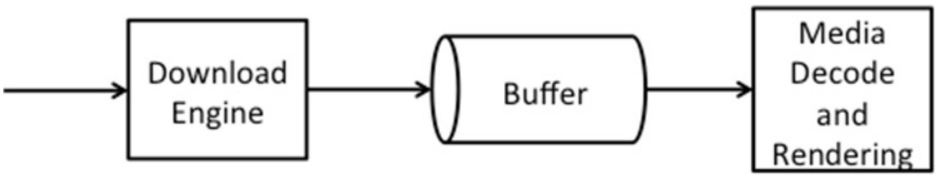


Figure 5 Simple Client Model

Assume that the client has access to an MPD and the MPD contains the parameters in Table 3, i.e. it consumes a dynamic service with fixed media presentation duration.

In addition in the following for simplicity it is assumed that the MPD only contains a single Period with period start time *PSwc*[*i*] and the MPD-URL does not include any fragment parameters according to section 4.3.3.5.

The following example client behaviour may provide a continuous streaming experience to the user:

1) The client parses the MPD, selects a collection of Adaptation Sets suitable for its environment based on information provided in each of the **`AdaptationSet`** elements.

2) Within each Adaptation Set it selects one Representation, typically based on the value of the `@bandwidth` attribute, but also taking into account client decoding and rendering capabilities.
3) The client creates a list of accessible Segments at least for each selected Representation taking into account the information in the MPD as documented in Table 3 and the current time *JOIN* in the client and in particular the segment closest to the live edge referred to the *live edge segment*. For details refer to section 4.3.4.2.
4) The client downloads the initialization segment of the selected Representations and then accesses the content by requesting entire Segments or byte ranges of Segments. Typically at any time download the next segment at the larger of the two: (i) completion of download of current segment or (ii) the Segment Availability Start Time of the next segment. Based on the buffer fullness and other criteria, rate adaptation is considered. Typically the first media segment that is downloaded is the *live edge segment*, but other decisions may be taken in order to minimize start-up latency. For details on initial buffering, refer to section 4.3.4.4.
5) According to Figure 5 media is fed into buffer and at some point in time, the decoding and rendering of the media is kicked off. The downloading and presentation is done for the selected Representation of each selected Adaptation. The synchronization is done using the presentation time in the Period as documented in section 4.3.2.2.9. For synchronized playout, the exact presentation times in the media shall be used.

   Once presentation has started, the playout process is continuous. The playout process expects media to be present in the buffer continuously. If the **MPD**`@suggestedPresentationDelay` is present, then this value may be used as the presentation delay *PD*. If the **MPD**`@suggestedPresentationDelay` is not present, but the client is expected to consume the service at the live edge, then a suitable presentation delay should be selected, typically between the value of `@minBufferTime` and the value of `@timeShiftBufferDepth`. It is recommended that the client starts rendering the first sample of the downloaded media segment *k* with earliest presentation time EPT(*k*) at $PSwc[i]$ + (EPT(*k*) - $o[r,i]$) + *PD*. For details on selecting and minimizing end-to-end latency as well as the start-up latency, see section 4.3.4.4.
6) The client may request Media Segments of the selected Representations by using the generated Segment list during the availability time window.

7) Once the presentation has started, the client continues consuming the media content by continuously requesting Media Segments or parts of Media Segments and playing content that according to the media presentation timeline. The client may switch Representations taking into updated information from its environment, e.g. change of observed throughput. In a straight-forward implementation, with any request for a Media Segment starting with a stream access point, the client may switch to a different Representation. If switching at a stream access point, the client shall switch seamlessly at such a stream access point.
8) With the wall-clock time *NOW* advancing, the client consumes the available Segments. As *NOW* advances the client possibly expands the list of *available* Segments for each Representation in the Period according to the procedures specified in 4.3.4.2.
9) Once the client is consuming media contained in the Segments towards the end of the announced media in the Representation, then either the Media Presentation is terminated, a new Period is started (see subsection 4.3.4.3) or the MPD needs to be refetched. For details on MPD updates and refetching, please refer to section 4.4.

4.3.4.2. Determining the Segment List

For a single Period content the client determines the available Segment List at time *NOW* according to section 4.3.2.2.7 taking into account the simplified offering in Table 4 as

- k1 = 1
- k2 = `ceil(PDURATION/SDURATION)`
- SAST[*k*] = `START` + `PSTART` + *k**`SDURATION` for k = 0, 1, ..., k2
- SAET[*k*] = SAST[*k*] + `TSB` + `SDURATION` for k = 1, ..., k2
- SAET[0] = SAET[k2]
- SD[*k*] = `SDURATION`
- URL[*k*] = `http://example.com/$RepresentationID$/k`
- k[*NOW*] = MIN(`floor` ((*NOW* - `START` - `PSTART`)/`SDURATION`), k2)
- k*[*NOW*] = MAX(k1, `floor`((*NOW* - `START` - `PSTART` - `TSB`)/`SDURATION`)

Note that if k[*NOW*] is *0*, then only the Initialization Segment is available.

4.3.4.3. Multi-Period Content

In an extension to the description in section 4.3.4.1 assume now that the client has access to an MPD and the MPD contains content with multiple Periods, for example following the parameters in Table 6. The start time of each Period is computed as period start time *PSwc*[*i*]. and the MPD-URL does not include any fragment parameters according to section 4.3.3.5.

In an extension of bullet 3 in section 4.3.4.1,

> the client creates a list of accessible Segments at least for each selected Representation taking into account the information in the MPD as documented in Table 3 and the current time *NOW* in the client and in particular the segment closest to the live edge referred to the *live edge segment*.

For this it needs to take into account the latest Period *i*[*NOW*]. The latest Period and the latest segment are obtained as follows with i* the index of the last Period.:

- if *NOW* <= PSwc[1]
  - no segment is yet available
- else if *NOW* > PSwc[i*]
  - the last one and the latest segment is available is k2[i*]
- else if *NOW* > PSwc[i*] `+ TSB`
  - no segment is available any more
- else if PSwc[1] < *NOW* <= PEwc[i*]
  - *i'* the such that PSwc[*i'*] < *NOW* <= PEwc[*i'*]
  - k[*NOW*] = MIN(`floor` ((*NOW* - PEwc[*i'*] - PSwc[*i'*])/`SDURATION[i']`), k2)
- Note again that if that if k[*NOW*] is *0*, then only the Initialization Segment is available. If the Period is not the first one, then the last available Media Segment is the last Media Segment of the previous Period.

In an extension of bullet 9 in section 4.3.4.1,

the client consumes media in one Period. Once the client is consuming media contained in the Segments towards the end of the announced media in the Representation, and the Representation is contained not in the last Period, then the DASH clients generally needs to reselect the Adaptation Sets and a Representation in same manner as described in bullet 1 and 2 in section 4.3.4.1. Also steps 3, 4, 5 and 6 need to be carried out at the transition of a Period. Generally, audio/video switching across period boundaries may not be seamless. According to ISO/IEC 23009-1, section 7.2.1, at the start of a new Period, the playout procedure of the media content components may need to be adjusted at the end of the preceding Period to match the *PeriodStart* time of the new Period as there may be small overlaps or gaps with the Representation at the end of the preceding Period. Overlaps (respectively gaps) may result from Media Segments with actual presentation duration of the media stream longer (respectively shorter) than indicated by the Period duration. Also in the beginning of a Period, if the earliest presentation time of any access unit of a Representation is not equal to the presentation time offset signalled in the `@presentationTimeOffset`, then the playout procedures need to be adjusted accordingly.

The client should play the content continuously across Periods, but there may be implications in terms of implementation to provide fully continuous and seamless playout. It may be the case that at Period boundaries, the presentation engine needs to be reinitialized, for example due to changes in formats, codecs or other properties. This may result in a re-initialization delay. Such a re-initialization delay should be minimized. If the Media Presentation is of type dynamic, the addition of the re-initialisation delay to the playout may result in drift between the encoder and the presentation engine. Therefore, the playout should be adjusted at the end of each Period to provide a continuous presentation without adding drift between the time documented in the MPD and the actual playout, i.e. the difference between the actual playout time and the Period start time should remain constant.

If the client presents media components of a certain Adaptation Set in one Period, and if the following Period has assigned an identical Asset Identifier, then the client should identify an associated Period and, in the absence of other information, continue playing the content in the associated Adaptation Set.

If furthermore the Adaptation Sets are period-continuous, i.e. the presentation times are continuous and this is signalled in the MPD, then the client shall seamlessly play the content across the Period boundary under the constraints in section 4.3.3.3.2. Most suitably the client may continue playing the Representation in the Adaptation Set with the same `@id`, but there is no guarantee that this Representation is available. In this case the client shall seamlessly switch to any other Representation in the Adaptation Set.

4.3.4.4. Joining, Initial Buffering and Playout Recommendations

4.3.4.4.1. General

A DASH client should start playout from:

- The time indicated by the MPD Anchor, if one is present
- The live edge, if there is no MPD Anchor and **`MPD`**`@type="dynamic"`.

4.3.4.4.2. Joining at the live edge

For joining at the live edge there are basically two high-level strategies:

- Every client participating in the service commits to the same presentation delay (PD) relative to the announced segment availability start time at start-up and in continuous presentation, possible using one suggested by the Content Provider and then attempts to minimise start-up latency and maintain the buffer. The content provider may have provided the **`MPD`**`@suggestedPresentationDelay` (`SPD`) or may have provided this value by other means outside the DASH formats.
- The client individually picks the presentation delay (PD) in order to maximize stable quality and does this dependent on its access, user preferences and other considerations.

In both cases the client needs to decide, which segment to download first and when to schedule the playout of the segment based on the committed PD.

A DASH client would download an available segment and typically render the earliest presentation time EPT(k) of the segment at $PSwc[i]$ + (EPT(k) - $o[r,i]$) + PD. As PD may be quite large, for example in order to provision for downloading in varying bitrate conditions, and if a segment is downloaded that was just made available it may result in larger start up delay.

Therefore, a couple of strategies may be considered as a tradeoff of for start-up delay, presentation delay and sufficient buffer at the beginning of the service, when joining at the live edge:

1. The client downloads the next available segment and schedules playout with delay `PD`. This maximizes the initial buffer prior to playout, but typically results in undesired long start-up delay.
2. The client downloads the latest available segment and schedules playout with delay `PD`. This provides large initial buffer prior to playout, but typically results in undesired long start-up delay.
3. The client downloads the earliest available segment that can be downloaded to schedules playout with delay `PD`. This provides a smaller initial prior to playout, but results in reasonable start-up delay. The buffer may be filled gradually by downloading later segments faster than their media playout rate, i.e. by initially choosing Representations that have lower bitrate than the access bandwidth.

In advanced strategies the client may apply also one or more of the following:

1. Actual rendering may start not with the sample of the earliest presentation time, but the one that matches as closely as possible *PSwc*[*i*] + (PT - *o*[*r,i*]) + *PD* equal to *NOW*.
2. The client may start rendering even if only a segment is downloaded partially.

4.3.4.5. Requirements and Recommendations

In summary, a client that access a dynamic MPD shall at least obey the following rules:

- The client shall be able to consume single Period and multi-Period content
- If multi-period content is offered in a seamless manner, the client shall play seamlessly across Period boundaries.

4.3.5. Additional DVB-DASH alignment aspects

For alignment with DVB-DASH [38], the following should be considered:

- Reasonable requirements on players around responding to response codes are provided in DVB DASH in section 10.8.6.
- Further guidelines on live edge aspects are provided in DVB DASH section 10.9.2.

DVB DASH also provides recommendations in order to apply weights and priorities to different networks in a multi Base URL offering in section 10.8.2.1.

4.4. Simple Live Service Offering including MPD Updates

4.4.1. Background and Assumptions

If many cases, the service provider cannot predict that an MPD that is once offered, may be used for the entire Media Presentations. Examples for such MPD changes are:

- The duration of the Media Presentation is unknown
- The Media Presentation may be interrupted for advertisements which requires proper splicing of data, for example by adding a Period
- Operational issues require changes, for example the addition of removal of Representations or Adaptation Sets.
- Operational problems in the backend, for example as discussed in section 4.8.
- Changes of segment durations, etc.

In this case the MPD typically only can describe a limited time into the future. Once the MPD expires, the service provider expects the client to recheck and get an updated MPD in order to continue the Media Presentation.

The main tool in MPEG-DASH is Media Presentation Description update feature as described in section 5.4 of ISO/IEC 23009-1. The MPD is updated at the server and the client is expected to obtain the new MPD information once the determined Segment List gets to an end.

If the MPD contains the attribute **`MPD`**`@minimumUpdatePeriod`, then the MPD in hand will be updated.

According to the clustering in section 4.2, we distinguish two different types of live service offerings:

- MPD controlled live service offering: In this case the DASH client typically frequently polls the MPD update server whether an MPD update is available or the existing MPD can still be used. The update frequency is controlled by MPD based on the attribute **`MPD`**`@minimumUpdatePeriod`. Such a service offering along with the client procedures is shown in section 4.4.2.
- MPD and segment controlled offerings. In this case the DASH client needs to parse segments in order to identify MPD validity expirations and updates on the MPD update server. MPD expiry events as described in section 5.10 of ISO/IEC 23009-1 "are pushed" to the DASH client as parts of downloaded media segments. This offering along with the client procedures is shown in section 4.5.

This section describes the first type of offering. In section 4.5 the MPD and segment controlled offerings are described. Under certain circumstances a service offering may be provided to both types of clients. An overview how such a service offering may be generated is shown in Annex A.

4.4.2. Preliminaries

4.4.2.1. MPD Information

As the MPD is typically updated over time on the server, the MPD that is accessed when joining the service as well as the changes of the MPD are referred to as MPD instances in the following. This expresses that for the same service, different MPDs exist depending on the time when the service is consumed.

Assume that an MPD instance is present on the DASH server at a specific wall-clock time *NOW*. For an MPD-based Live Service Offering, the MPD instance may among others contain information as available in Table 8. Information included there may be used to compute a list of announced Segments, Segment Availability Times and URLs.

Table 8 – Information related to Live Service Offering with MPD-controlled MPD Updates

MPD Information	Status	Comment

MPD@type	mandatory, set to "dynamic"	the type of the Media Presentation is dynamic, i.e. Segments get available over time.
MPD@availabilityStartTime	mandatory	the start time is the anchor for the MPD in wall-clock time. The value is denoted as *AST*.
MPD@minimumUpdatePeriod	mandatory	this field is mandatory except for the case where the **MPD**@mediaPresentationDuration is present. However, such an MPD falls then in an instance as documented in section 4.3.
Period@start	mandatory	the start time of the Period relative to the MPD availability start time. The value is denoted as *PS*.
SegmentTemplate@media	mandatory	the template for the Media Segment
SegmentTemplate@startNumber	optional default	the number of the first segment in the Period. The value is denoted as *SSN*.
SegmentTemplate@duration	exactly one of **SegmentTemplate**@duration or **SegmentTemplate.SegmentTimeline** must be present	the duration of each Segment in units of a time. The value divided by the value of @timescale is denoted as *MD*[*k*] with k=1, 2, ... The segment timeline may contain some gaps.
SegmentTemplate.SegmentTimeline		

4.4.2.2. Segment Information Derivation

Based on an MPD instance including information as documented in Table 8 and available at time *NOW* on the server, a DASH client may derive the information of the list of Segments for each Representation in each Period.

If the Period is the last one in the MPD and the **MPD**@minimumUpdatePeriod is present, then the time *PEwc*[*i*] is obtained as the sum of *NOW* and the value of **MPD**@minimumUpdatePeriod.

Note that with the MPD present on the server and *NOW* progressing, the Period end time is extended. This issue is the only change compared to the segment information generation in section 4.3.2.2.

4.4.2.3. Some Special Cases

If the **MPD**@minimumUpdatePeriod is set to 0, then the MPD documents all available segments on the server. In this case the @r count may be set accurately as the server knows all available information.

4.4.3. Service Offering Requirements and Guidelines

4.4.3.1. General

The same service requirements as in section 4.3.3.1 hold for any time *NOW* the MPD is present on the server with the interpretation that the Period end time *PEwc*[*i*] of the last Period is obtained as the sum of *NOW* and the value of **MPD**@minimumUpdatePeriod.

In order to offer a simple live service with unknown presentation end time, but only a single Period and the following details are known in advance,

- start at wall-clock time START,
- location of the segments for each Representation at " http://example.com/$RepresentationID$/$Number$",

a service provider may offer an MPD with values according to Table 9.

Table 9 – Basic Service Offering with MPD Updates

MPD Information	Value
MPD@type	dynamic
MPD@availabilityStartTime	START
MPD@publishTime	PUBTIME1
MPD@minimumUpdatePeriod	MUP
MPD.BaseURL	"http://example.com/"
Period@start	PSTART
SegmentTemplate@media	"$RepresentationID$/$Number$"
SegmentTemplate@startNumber	1
SegmentTemplate@duration	SDURATION

According to the work-flow shown in Annex B,

- the MPD is generated and published prior to time START such that DASH clients may access it prior to the start of the Media Presentation. The MPD gets assigned a publish time PUBTIME1, typically a value that is prior to START + PSTART
- no redundant tools are considered.
- the encoder and the segmenter generate segments of duration SDURATION and publish those on the origin server, such that they are available at URL[*k*] latest at their announced segment availability start time SAST[*k*].

Based on the details in section 4.3.2.2 and 4.4.2.2, the Segment Information can be derived at each time *NOW* by determining the end time of the Period *PEwc*[1] = *NOW* + MUP.

The service provider may leave the MPD unchanged on the server. If this is the case the Media Presentation may be terminated with an updated MPD that

- adds the attribute `MPD@mediaPresentationDuration` with value `PDURATION`
- removes the attribute `MPD@minimumUpdatePeriod`
- changes the `MPD@publishTime` attribute to `PUBTIME2`

The MPD must be published latest at the end of the Media Presentation minus the value of `MUP`, i.e. `PUBTIME2` <= `START` + `PSTART` + `PDURATION` - `MUP`.

The minimum update period may also be changed during an ongoing Media Presentation. Note that as with any other change to the MPD, this will only be effective with a delay in media time of the value of the previous `MUP`.

The principles in this document also holds for multi-period content, for which an MPD update may add a new Period. In the same way as for signalling the end of the Media Presentation, the publish time of the updated MPD with the new period needs to be done latest at the start of the new Period minus the value of the `MPD@minimumUpdatePeriod` attribute of the previous MPD.

Track fragment decode times should not roll over and should not exceed 2^{53} (due to observed limitations in ECMAScript). Two options may be considered:

- the timescale value should be selected that the above mentioned issues are avoided. 32 bit timescales are preferable for installed-base of browsers.
- if large track timescale values are required and/or long-lasting live sessions are setup, this likely requires the use of 64 bit values. Content authors should use 64 bit values for track fragment decode times in these cases, but should not exceed to 2^{53} to avoid truncation issues.

4.4.3.2. Setting the Minimum Update Period Value

Setting the value of the minimum update period primarily affects two main service provider aspects: A short minimum update period results in the ability to change and announce new content in the MPD on shorter notice. However, by offering the MPD with a small minimum update period, the client requests an update of the MPD more frequently, potentially resulting in increased uplink and downlink traffic.

A special value for the minimum update period is 0. In this case, the end time of the period is the current time *NOW*. This implies that all segments that are announced in the MPD are actually available at any point in time. This also allows changing the service provider to offer changes in the MPD that are instantaneous on the media timeline, as the client, prior for asking for a new segment, has to revalidate the MPD.

4.4.3.3. Permitted Updates in an MPD

According to section 5.4 of ISO/IEC 23009-1, when the MPD is updated

- the value of `MPD@id`, if present, shall be the same in the original and the updated MPD;

- the values of any **Period**@id attributes shall be the same in the original and the updated MPD, unless the containing **Period** element has been removed;
- the values of any **AdaptationSet**@id attributes shall be the same in the original and the updated MPD unless the containing **Period** element has been removed;
- any Representation with the same @id and within the same Period as a Representation appearing in the previous MPD shall provide functionally equivalent attributes and elements, and shall provide functionally identical Segments with the same indices in the corresponding Representation in the new MPD.

In addition, updates in the MPD only extend the timeline. This means that information provided in a previous version of the MPD shall not be invalidated in an updated MPD. For failover cases, refer to section 4.8.

In order to make the MPD joining friendly and to remove data that is available in the past, any segments that have fallen out of the time shift buffer may no longer be announced in the MPD. In this case, the Period start may be moved by changing one or both, **MPD**@availabilityStartTime and **Period**@start. However, this requires that the @startNumber, @presentationTimeOffset and **S** values need to be updated such that the Segment Information according to section 4.3.2.2.6 is not modified over an MPD update.

If Representations and Adaptations Sets are added or removed or the location of the Segments is changed, it is recommended to update the MPD and provide Adaptation Sets in a period-continuous manner as defined in section 4.3.3.3.2.

4.4.3.4. Usage of Segment Timeline

If the Segment Timeline is used and @minimumUpdatePeriod greater than 0, then

- the operation as described in section 4.3.3.4 applies, and for all Representations that use the Segment Timeline:
  - the @r value of the last **S** element of the last regular Period shall be a negative value,
  - only $Number$ template shall be used,
- an MPD may be published for which the additional **S** elements are added at the end. An addition of such **S** element shall be such that clients that have not updated the MPD can still generate the Segment Information based on the previous MPD up to the Period end time. Note that this may lead that such clients have a different segment availability time, but the availability time may be corrected once the MPD is updated.

An example for such an offering is shown in Table 10 where the RVALUE needs to be increased by 1 for each newly published segment.

Table 10 – Service Offering with Segment Timeline and MUP greater than 0

MPD Information	Value
MPD@type	dynamic
MPD@availabilityStartTime	START

MPD@publishTime	PUBTIME1
MPD@minimumUpdatePeriod	MUP > 0
MPD.BaseURL	"http://example.com/"
Period@start	PSTART
SegmentTemplate@media	"$RepresentationID$/$Time$"
SegmentTemplate@d	SDURATION
SegmentTemplate.SegmentTimeline.S@r	-1

4.4.3.5. Last Segment Message

If the @segmentProfiles contains the 'lmsg' brand for a certain Representation, then the 'lmsg' brand for signaling the last segment shall be applied for any content with **MPD**@minimumUpdatePeriod present and the **MPD**@type="dynamic".

DASH clients operating based on such an MPD and consuming the service at the live edge typically need to request a new MPD prior to downloading a new segment. However, in order to minimise MPD requests and resulting traffic load, the client may use one or more of the following optimisations:

- If the client fetches the MPD using HTTP, the client should use conditional GET methods as specified in RFC 2616 [22], clause 9.3 to reduce unnecessary network usage in the downlink.
- If the @segmentProfiles contains the 'lmsg' brand clients may also rely on the 'lmsg' message and request a new MPD only in case a segment is received with an 'lmsg' brand. Otherwise the client may use template constructions to continue determining the URL and the segment availability start time of segments.

If the attribute **MPD**@minimumUpdatePeriod is set to a value greater than 0 then all Segments with availability start time less than the sum of the request time and the value of the **MPD**@minimumUpdatePeriod will eventually get available at the advertised position at their computed segment availability start time. Note that by providing a **MPD**@minimumUpdatePeriod is set to a value greater than 0, DASH servers reduce the polling frequency of clients, but at the same time cannot expect that clients will request an updated MPD to be informed on changes in the segment URL constructions, e.g. at the start of a new Period.

4.4.4. MPD-based Live Client Operation based on MPD

In an extension to the description in section 4.3.4.1 and section 4.3.4.3, the client now has access to an MPD and the MPD contains the **MPD**@minimumUpdatePeriod, for example following the parameters in Table 9. The start time of each Period is computed as period start time *PSwc*[*i*] and the MPD-URL does not include any fragment parameters according to section 4.3.3.5.

The client fetches an MPD with parameters in Table 8 access to the MPD at time FetchTime, at its initial location if no **MPD.Location** element is present, or at a location specified in any present **MPD.Location** element. FetchTime is defined as the time at which the server processes the request for the MPD from the client. The client typically should not use the time at which it actually successfully received the MPD, but should take into account delay due to MPD delivery and processing. The fetch is considered successful either if the client obtains an updated MPD or the client verifies that the MPD has not been updated since the previous fetching.

If the client fetches the MPD using HTTP, the client should use conditional GET methods as specified in RFC 2616 [9], clause 9.3 to reduce unnecessary network usage in the downlink.

In an extension of bullet 3 in section 4.3.4.1 and section 4.3.4.3

> the client creates a list of accessible Segments at least for each selected Representation taking into account the information in the MPD as documented in Table 8 and the current time *NOW* by using the Period end time of the last Period as `FetchTime + MUP`.

In an extension of bullet 9 in section 4.3.4.1 and section 4.3.4.3,

> the client consumes media in last announced Period. Once the client is consuming media contained in the Segments towards the end of the announced Period, i.e. requesting segments with segment availability start time close to the validity time of the MPD defined as `FetchTime + MUP`, them, then the DASH client needs to fetch an MPD at its initial location if no **`MPD.Location`** element is present, or at a location specified in any present **`MPD.Location`** element.
>
> If the client fetches the updated MPD using HTTP, the client should use conditional GET methods as specified in RFC 2616, clause 9.3 to reduce unnecessary network usage in the downlink.
>
> The client parses the MPD and generates a new segment list based on the new `FetchTime` and `MUP` of the updated MPD. The client searches for the currently consumed Adaptation Sets and Representations and continues the process of downloading segments based on the updated Segment List.

4.5. MPD and Segment-based Live Service Offering

4.5.1. Preliminaries

4.5.1.1. MPD Information

In order to offer a service that relies on both, information in the MPD and in Segments, the Service Provider may announce that Segments contains inband information. An MPD as shown in Table 9 provides the relevant information. In contrast to the offering in Table 6, the following information is different:

- The **`MPD`**`@minimumUpdatePeriod` is present but is recommended to be set to 0 in order to announce instantaneous segment updates.
- The **`MPD`**`@publishTime` is present in order to identify different versions of MPD instances.
- all Representations of all audio Adaptation Sets or if audio is not present, of all video Adaptation Sets, shall contain an **`InbandEventStream`** element with

@scheme_id_uri = "urn:mpeg:dash:event:2012" and @value either set to 1 or set to 3. The **InbandEventStream** element with @scheme_id_uri = "urn:mpeg:dash:event:2012" and @value either set to 1 or set to 3 may be present in all Representations of all Adaptation Sets.

- **InbandEventStream** element with @scheme_id_uri = "urn:mpeg:dash:event:2012" and @value either set to 1 or set to 3 shall only be signaled on Adaptation Set level.

The information included there may be used to compute a list of announced Segments, Segment Availability Times and URLs.

Table 11 – Service Offering with MPD and Segment-based Live Services

MPD Information	Status	Comment
MPD@type	mandatory, set to "dynamic"	the type of the Media Presentation is dynamic, i.e. Segments get available over time.
MPD@publishTime	mandatory	specifies the wall-clock time when the MPD was generated and published at the origin server. MPDs with a later value of @publishTime shall be an update as defined in 5.4 to MPDs with earlier @publishTime.
MPD@availabilityStartTime	mandatory	the start time is the anchor for the MPD in wall-clock time. The value is denoted as *AST*.
MPD@minimumUpdatePeriod	mandatory	recommended/mandate to be set to 0 to indicate that frequent DASH events may occur
Period@start	mandatory	the start time of the Period relative to the MPD availability start time. The value is denoted as *PS*.
AdaptationSet.InbandEventStream	mandatory	if the @schemeIDURI is urn:mpeg:dash:event:2014 and the @value is 1, 2 or 3, then this described

		an Event Stream that supports extending the validity of the MPD.
SegmentTemplate@media	mandatory	the template for the Media Segment
SegmentTemplate@startNumber	optional default	The number of the first segment in the Period. The value is denoted as *SSN*.
SegmentTemplate@duration	exactly one of **SegmentTemplate**@duration or **SegmentTemplate.SegmentTimeline** must be present	the duration of each Segment in units of a time. The value divided by the value of @timescale is denoted as *MD*[*k*] with k=1, 2, ... The segment timeline may contain some gaps.
SegmentTemplate.SegmentTimeline		

4.5.1.2. Segment Information Derivation

Based on an MPD instance including information as documented in Table 8 and available at time *NOW* on the server, a DASH client may derive the information of the list of Segments for each Representation in each Period.

If the Period is the last one in the MPD and the **MPD**@minimumUpdatePeriod is present, then the time *PEwc*[*i*] is obtained as the sum of *NOW* and the value of **MPD**@minimumUpdatePeriod.

Note that with the MPD present on the server and *NOW* progressing, the Period end time is extended. This issue is the only change compared to the segment information generation in section 4.3.2.2.

If the **MPD**@minimumUpdatePeriod is set to 0, then the MPD documents all available segments on the server. In this case the @r count may be set accurately as the server knows all available information.

4.5.2. Service Offering Requirements and Guidelines

4.5.2.1. Background

In section 5.10 of ISO/IEC 23009-1, section 5.10, DASH events are defined. For service offerings based on the MPD and segment controlled services, the DASH events specified in section 5.10.4 may be used. Background is provided in the following.

DASH specific events that are of relevance for the DASH client are signalled in the MPD. The URN "urn:mpeg:dash:event:2012" is defined to identify the event scheme defined in Table 10.

Table 12 `InbandEventStream@value` attribute for scheme with a value `"urn:mpeg:dash:event:2012"`

@value	Description
1	indicates that MPD validity expiration events as defined in 5.10.4.2 are signalled in the Representation. MPD validity expiration is signalled in the event stream as defined in 5.10.4.2 at least in the last segment with earliest presentation time smaller than the event time.
2	indicates that MPD validity expiration events as defined in 5.10.4.3 are signalled in the Representation. MPD validity expiration is signalled in the event stream as defined in 5.10.4.2 at least in the last segment with earliest presentation time smaller than the event time. In addition the message includes an MPD Patch as defined in 5.10.4.3 in the **`message_data`** field.
3	indicates that MPD validity expiration events as defined in 5.10.4.3 are signalled in the Representation. MPD validity expiration is signalled in the event stream as defined in 5.10.4.2 at least in the last segment with earliest presentation time smaller than the event time. In addition the message includes a full MPD as defined in 5.10.4.4 in the message_data field.

Note: DVB DASH specification [38] does not include the value 3.

MPD validity expiration events provide the ability to signal to the client that the MPD with a specific publish time can only be used up to a certain media presentation time.

Figure 4 shows an example for MPD validity expiration method. An MPD signals the presence of the scheme in one or several Representations. Once a new MPD gets available, that adds new information not present in the MPD with `@publishTime="2012-11-01T09:06:31.6"`, the expiration time of the current MPD is added to the segment by using the emsg box. The information may be present in multiple segments.

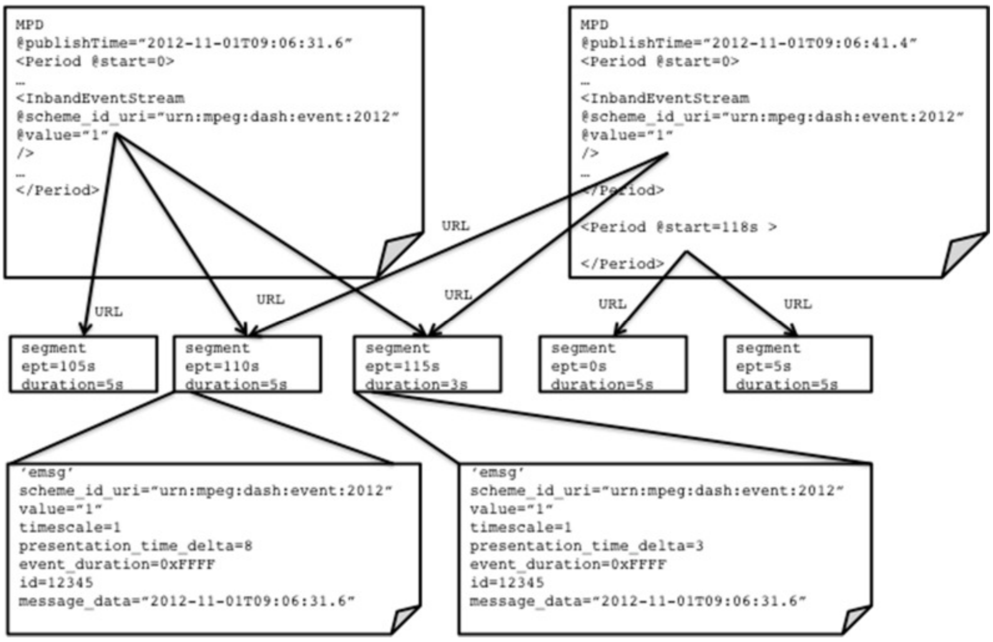


Figure 4 Example for MPD validity expiration to signal new Period

If the `scheme_id_uri` is set to "`urn:mpeg:dash:event:2012`" and the `value` is set to 1, then the fields in the event message box document the following:

- the `message_data` field contains the publish time of an MPD, i.e. the value of the **MPD**`@publishTime`.
- The media presentation time beyond the event time (indicated time by `presentation_time_delta`) is correctly described only by MPDs with publish time greater than indicated value in the `message_data` field.
- the event duration expresses the remaining duration of Media Presentation from the event time. If the event duration is 0, Media Presentation ends at the event time. If 0xFFFF, the media presentation duration is unknown. In the case in which both `presentation_time_delta` and `event_duration` are zero, then the Media Presentation is ended.

This implies that clients attempting to process the Media Presentation at the event time or later are expected to operate on an MPD with a publish time that is later than the indicated publish time in this box.

Note that event boxes in different segments may have identical `id` fields, but different values for `presentation_time_delta` if the earliest presentation time is different across segments.

4.5.2.2. Service Offering

A typical service offering with an Inband event stream is provided in Table 11. In this case the MPD contains information that one or multiple or all Representations contain information that the Representation contains an event message box flow in order to signal MPD validity expirations. The **MPD**`@publishTime` shall be present.

Table 13 – Basic Service Offering with Inband Events

MPD Information	Value
MPD@type	dynamic
MPD@availabilityStartTime	START
MPD@publishTime	PUBTIME1
MPD@minimumUpdatePeriod	MUP
MPD.BaseURL	"http://example.com/"
Period@start	PSTART
InbandEvent-Stream@scheme_id_URI	urn:mpeg:dash:event:2012
InbandEventStream@value	1 or 3
SegmentTemplate@duration	SDURATION

For a service offering based on MPD and segment-based controls, the DASH events shall be used to signal MPD validity expirations.

In this case the following shall apply:

- at least all Representations of all audio Adaptation Sets shall contain an **InbandEventStream** element with scheme_id_uri = "urn:mpeg:dash:event:2014" and @value either set to 1 or set to 3.
- for each newly published MPD, that includes changes that are not restricted to any of the following (e.g. a new Period):
    - The value of the **MPD**@minimumUpdatePeriod is changed,
    - The value of a **SegmentTimeline.S@**r has changed,
    - A new **SegmentTimeline.S** element is added
    - Changes that do not modify the semantics of the MPD, e.g. data falling out of the timeshift buffer can be removed, changes to service offerings that do not affect the client, etc.

  the following shall be done

    - a new MPD shall be published with a new publish time **MPD**@publishTime
    - an 'emsg' box shall be added to each segment of each Representation that contains an **InbandEventStream** element with
        - scheme_id_uri = "urn:mpeg:dash:event:2012"
        - @value either set to 1 or set to 3
        - If @value set to 1 or 3
            - the value of the **MPD**@publishTime of the previous MPD as the message_data

In addition, the following recommendations should be taken into account: All Representations of at least one media type/group contain an **InbandEventStream** element with scheme_id_uri = "urn:mpeg:dash:event:2012" and @value either set to 1 or set to 3.

4.5.3. Client Requirements and Guidelines

4.5.3.1. Introduction

A DASH client is guided by the information provided in the MPD. An advanced client model is shown in Figure 6. In contrast to the client in section 4.4.3.5, the advanced client requires parsing of segments in order to determine the following information:

- to expand the Segment List, i.e. to generate the Segment Availability Start Time as well as the URL of the next Segment by parsing the Segment Index.
- to update the MPD based on Inband Event Messages using the `'emsg'` box with `scheme_id_uri="urn:mpeg:dash:event:2012"` and `@value` either set to 1 or set to 3.

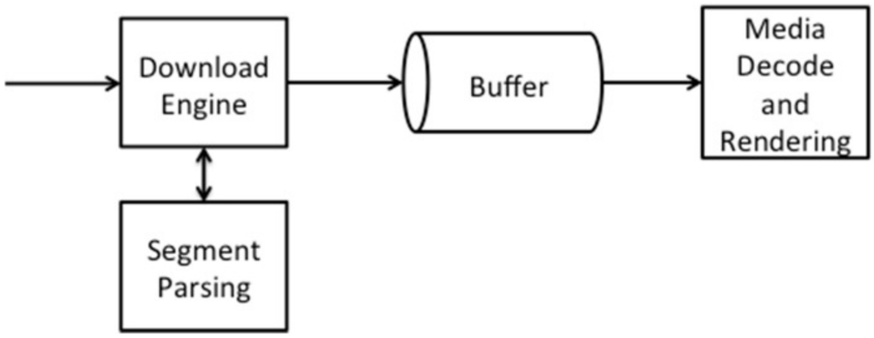


Figure 6 Advanced Client Model

Assumes that the client has access to an MPD and the MPD contains the mandatory parameters in Table 9, i.e., it contains the following information:

- **MPD@**`minimumUpdatePeriod` is set to 0
- **MPD**@publishTime is included and the value is set to PUBTIME
- At least on Representation is present that contains an **InbandEventStream** element with `scheme_id_uri="urn:mpeg:dash:event:2012"` and `@value` either set to 1 or set to 3.
- Either the `@duration` or **SegmentTimeline** for the Representation is present.

In an extension of bullet 7, 8 and 9 in section 4.3.4.1 and section 4.3.4.3, the following example client behaviour may provide a continuous streaming experience to the user as documented in the following.

4.5.3.2. MPD Validity expiration and Updates

The DASH client shall download at least one Representation that contains **InbandEventStream** element with `scheme_id_uri = "urn:mpeg:dash:event:2012"` and `@value` either set to 1 or set to 3. It shall parse the segment at least up to the first `'moof'` box. The DASH client shall parse the segment information and extract the following values:

- `ept` the earliest presentation time of the media segment
- `dur` the media presentation duration of the media segment

If an 'emsg' is detected `scheme_id_uri = "urn:mpeg:dash:event:2012"` and `@value` either set to 1 or set to 3, the DASH client shall parse the segment information and extract the following values:

- `emsg.publish_time` the publish time documented in the message data of the emsg, either directly or from the patch.
- `emsg.ptd` the presentation time delta as documented in the emsg.
- `emsg.ed` the event duration as documented in the emsg

After parsing, the Segment is typically forwarded to the media pipeline if it also used for rendering, but it may either be dumped (if the Representation is only used to access the DASH event, such as muted audio).

If no 'emsg' validity expiration event is included, then

- the current MPD can at least be used up to a media presentation time `ept` + `dur`

else if an 'emsg' validity expiration event is included, then

- the MPD with publish time equal to `emsg.publish_time` can only be used up to a media presentation time `ept` + `emsg.ptd`. Note that if `dur` > `emsg.ptd`, then the Period is terminated at `ept` + `emsg.ptd`.
- any MPD with publish time greater than `emsg.publish_time` can at least be used up to a media presentation time `ept` + `emsg.ptd`
- prior to generating a segment request with earliest presentation time greater than `ept` + `emsg.ptd`, the MPD shall either
  - o be refetched and updated by the client.
  - o or if `@value=3`, it may be used as included in the message.

NOTE: The DVB DASH profile [38] explicitly forbids downloading a Representation solely to gain access to an Inband Event Stream contained within it. For reference, the relevant part of the DVB DASH specification is section 9.1.6.

4.5.3.3. Extended Segment Information

The DASH client shall download the selected Representation and shall parse the segment at least up to the first `'moof'` box. The DASH client shall parse the segment information and extract the following values:

- `ept` the earliest presentation time of the media segment
  - o if the Segment Index is present use the Segments Index
  - o if not use the `baseMediaDecodeTime` in `'tfdt'` of the first movie fragment as the earliest presentation time
- `dur` the media presentation duration of the media segment
  - o if the Segment Index is present use the Segments Index
  - o if not use aggregated sample durations of the first movie fragment as the duration

Using this information, the DASH client should extend the Segment information and, if present the Segment Timeline with the information provided in the Segment. This information can then be used to generate the URL of the next Segment of this Representation. This avoids that the client fetches the MPD, but uses the information of the Segment Timeline. However, in any doubt of the information, for example if a new Adaptation Set is selected, or if Segments or lost, or in case of other operational issues, the DASH client may refetch the MPD in order to obtain the complete information from the MPD.

4.6. Provisioning of Live Content in On-Demand Mode

4.6.1. Scenario

A common scenario for DASH distribution results that a live generated service is also made available for On-Demand offering after the live program is completed. The typical scenario is as follows:

- The Segments as generated for the live service are also used for the On-Demand case. This avoids reformatting and also permits to reuse the Segments that are already cached.
- The MPD is modified to reflect that the content is available as On-Demand now.
- Problems that results from live delivery may be solved, e.g. variable segment durations, or issues of segment unavailability.
- The content may be augmented with ads.
- The content may be trimmed from a longer, e.g. 24/7 stream, at the beginning and/or end.

4.6.2. Content Offering Requirements and Recommendations

In order to provide live content as On-Demand in the above scenario, the following is recommended:

- The same Segments as generated for the live distribution are reused also for static distribution.
- Typically, the Segments also will have the same URL in order to exploit caching advantages.
- An MPD should be generated latest at the end of the live session, but also may be created during an ongoing live session to document a certain window of the program that is offered for On-Demand.
- A new MPD is generated that should contain the following information
    - o The **MPD**@type is set to `static`.
    - o The **MPD**@availabilityStartTime may be set to any time in the past, for example the time of the original "live" MPD may reused.
    - o As profile, the simple live profile may be used
    - o The attributes @timeShiftBufferDepth and @minimumUpdatePeriod are not present (in contrast to the live MPD) and a @mediaPresentationDuration attribute is added.
    - o The window offered by the MPD is expressed by appropriately setting the **Period**@start value (including the presentation time offset and the start

number) and the `@mediaPresentationDuration` attribute. The wall-clock time should be maintained by offsetting the **Period**`@start` without changing the **MPD**`@availabilityStartTime`.

- Content may be offered in the same Period structure as for live or in a different one.
  - If Periods are continuous, it is preferable to remove the Period structure.
  - If new Periods are added for Ad Insertion, the Periods preferably be added in a way that they are at Segment boundaries.
- Independent whether the `@duration` attribute or the **SegmentTimeline** element was used for the dynamic distribution, the static distribution version may have a **SegmentTimeline** with accurate timing to support seeking and to possibly also signal any gaps in the Segment timeline. To obtain the accurate timeline, the segments may have to be parsed (at least up to the tfdt) to extract the duration of each Segment.
- The same templating mode as used in the live service should also be used for static distribution.
- DASH Event streams (i.e., MPD validity expirations) should not be present in the MPD.

4.6.3. Client Behavior

For a DASH client, there is basically no difference on whether the content was generated from a live service or the content is provided as On-Demand. However, there are some aspects that may be "left-overs" from a live service distribution that a DASH client should be aware of:

- The Representations may show gaps in the Segment Timeline. Such gaps should be recognized and properly handled. For example a DASH client may find a gap only in one Representation of the content and therefore switches to another Representation that has no gap.
- The DASH client shall ignore any possibly present DASH Event boxes (e.g., MPD validity expirations) for which no Inband Event Stream is present in the MPD.

4.7. Availability Time Synchronization between Client and Server

4.7.1. Background

According to ISO/IEC 23009-1 [1] and section 4.3, in order to properly access MPDs and Segments that are available on origin servers or get available over time, DASH servers and clients should synchronize their clocks to a globally accurate time standard.

Specifically Segment Availability Times are expected to be wall-clock accurately announced in the MPD and the client needs to have access to the same time base as the MPD generation in order to enable a proper service. In order to ensure this, this section provides server and client requirements to ensure proper operation of a live service.

4.7.2. Service Provider Requirements and Guidelines

If the Media Presentation is dynamic or if the **MPD**@availabilityStartTime is present then the service shall provide a Media Presentation as follows:

- The segment availability times announced in the MPD should be generated from a device that is synchronized to a globally accurate timing source, preferably using NTP.
- The MPD should contain at least one **UTCTiming** element with @schemeIdURI set to one of the following:
    - urn:mpeg:dash:utc:ntp:2014
    - urn:mpeg:dash:utc:http-head:2014
    - urn:mpeg:dash:utc:http-xsdate:2014
    - urn:mpeg:dash:utc:http-iso:2014
    - urn:mpeg:dash:utc:http-ntp:2014
- If the MPD does not contain any element **UTCTiming** then the segments shall be available latest at the announced segment availability time using a globally accurate timing source.
- If the MPD contains an element **UTCTiming** then
    - the announced timing information in the **UTCTiming** shall be accessible to the DASH client, and
    - the segments shall be available latest at the announced segment availability time in the MPD for any device that uses one of announced time synchronization methods at the same time.

If urn:mpeg:dash:utc:http-head:2014 is used, then the server specified in the @value attribute of the **UTCTiming** element should be the server hosting the DASH segments such that with each request the Date general-header field in the HTTP header (see RFC2616 [18], section 14.18) can be used by the client to maintain synchronization.

Note that in practical deployments segment availability may be an issue due to failures, losses, outages and so on. In this case the Server should use methods as defined in section 4.8 to inform DASH clients about potential issues on making segments available.

A leap second is added to UTC every 18 months on average. A service provider should take into account the considerations in RFC 7164 [46].

The MPD time does not track leap seconds. If these occur during a live service they may advance or retard the media against the real time.

4.7.3. Client Requirements and Guidelines

If the Media Presentation is dynamic or if the **MPD**@availabilityStartTime is present then client should do the following:

- If the MPD does not contain any element **UTCTiming** it should acquire an accurate wall-clock time from its system. The anticipated inaccuracy of the timing source should be taken into account when requesting segments close to their segment availability time boundaries.

- If the MPD contains one or several elements **`UTCTiming`** then the client should at least use one of the announced timing information in the **`UTCTiming`** to synchronize its clock. The client must not request segments prior to the segment availability start time with reference to any of the chosen **`UTCTiming`** methods.

    Note: The DVB DASH [38] spec requires support for `http-xsdate` and `http-head` but allows content providers to include others in addition, and allows clients to choose others in preference if they wish. For details, refer to section 4.7 of the DVB DASH specification.

- The client may take into account the accuracy of the timing source as well as any transmission delays if it makes segment requests.
- Clients shall observe any difference between their time zone and the one identified in the MPD, as MPDs may indicate a time which is not in the same timezone as the client.
- If the client observes that segments are not available at their segment availability start time, the client should use the recovery methods defined in section 4.8.
- Clients should not access the **`UTCTiming`** server more frequently than necessary.

4.8. Robust Operation

4.8.1. Background

In order to support some of the advanced use cases documented in section 2, robust service offerings and clients are relevant. This document lists the relevant ones.

4.8.2. Tools for Robust Operations

4.8.2.1. General Robustness

General Guidelines in ISO/IEC 23009-1 [1] DASH spec in A.7:

- The DASH access client provides a streaming service to the user by issuing HTTP requests for Segments at appropriate times. The DASH access client may also update the MPD by using HTTP requests. In regular operation mode, the server typically responds to such requests with status code 200 OK (for regular GET) or status code 206 Partial Content (for partial GET) and the entity corresponding to the requested resource. Other Successful 2xx or Redirection 3xx status codes may be returned.
- HTTP requests may result in a Client Error 4xx or Server Error 5xx status code. Some guidelines are provided in this subclause as to how an HTTP client may react to such error codes.
- If the DASH access client receives an HTTP client or server error (i.e. messages with 4xx or 5xx error code), the client should respond appropriately (e.g. as indicated in RFC 2616) to the error code. In particular, clients should handle redirections (such as 301 and 307) as these may be used as part of normal operation.
- If the DASH access client receives a repeated HTTP error for the request of an MPD, the appropriate response may involve terminating the streaming service.

- If the DASH access client receives an HTTP client error (i.e. messages with 4xx error code) for the request of an Initialization Segment, the Period containing the Initialization Segment may not be available anymore or may not be available yet.
- Similarly, if the DASH access client receives an HTTP client error (i.e. messages with 4xx error code) for the request of a Media Segment, the requested Media Segment may not be available anymore or may not be available yet. In both these case the client should check if the precision of the time synchronization to a globally accurate time standard is sufficiently accurate. If the clock is believed accurate, or the error re-occurs after any correction, the client should check for an update of the MPD.
- Upon receiving server errors (i.e. messages with 5xx error code), the client should check for an update of the MPD. If multiple **`BaseURL`** elements are available, the client may also check for alternative instances of the same content that are hosted on a different server.

4.8.3. Synchronization Loss of Segmenter

In order to address synchronization loss issues at the segmenter, the following options from the DASH standard should be considered with preference according to the order below:

1. The server is required to always offer a conforming media stream. In case the input stream or encoder is lost, the content author may always add dummy content. This may be done using a separate Period structure and is possible without any modifications of the standard.

2. Short Periods as included Cor.1 of the second edition of ISO/IEC 23009-1. Short Periods may be added that contain both **`Period`**`@start` and **`Period`**`@duration`. This expresses that for this Period no media is present at least for the time as expressed by the `@duration` attribute. Such Periods should only be used if Media Presentation author is experiencing issues in generating media, e.g. due to failures of a live feed. The MPD is updated using the @minimumUpdatePeriod, i.e. the timeline is progressing. This permits server to signal that there is an outage of media generation, but that the service is continuing. It is then up to the client to take appropriate actions.

4.8.4. Encoder Clock Drift

In order to support robust offering even under encoder drift circumstances, the segmenter should avoid being synced to the encoder clock. In order to improve robustness, in the case of an MPD-based offering Periods should be added in a period continuous manner. In the case of MPD and segment-based control, the producer reference box should be added to media streams in order for the media pipeline to be aware of such drifts. In this case the client should parse the segment to obtain this information.

4.8.5. Segment Unavailability

To address signaling of segment unavailability between the client and server and to indicate the reason for this, it is recommended to use regular 404s. In addition, unless a UTC

Timing has been defined prior in the MPD, the Date-Header specifying the time of the server should be used. In this case, the DASH client, when receiving a 404, knows that if its time is matching the Date Header, then the loss is due to a segment loss.

4.8.6. Swapping across Redundant Tools

To enable swapping across redundant tools doing hot and warm swaps, the following should be considered

1. the content author is offering the service redundant to the client (for example using multiple BaseURLs) and the client determines the availability of one or the other. This may be possible under certain circumstances
2. Periods may be inserted at a swap instance in order to provide the new information after swap. If possible, the offering may be continuous, but the offering may also be non-continuous from a media time perspective.
3. A completely new MPD is sent that removes all information that was available before any only maintains some time continuity. However, this tool is not fully supported yet in any DASH standard and not even considered.

There is a clear preference for the bullets above in their order 1, 2 and 3 as the service continuity is expected to be smoother with higher up in the bullet list. At the same time, it may be the case that the failure and outages are severe and only the third option may be used.

4.8.7. Service Provider Requirements and Guidelines

The requirements and guidelines in subsections 8.2 to 8.6 shall be followed.

4.8.8. Client Requirements and Guidelines

The client shall implement proper methods to deal with service offerings provided in section 8.2 to 8.6.

4.9. Interoperability Aspects

4.9.1. Introduction

In order to provide interoperability based on the tools introduce in this section a restricted set of interoperability points are defined.

4.9.2. Simple Live Operation

4.9.2.1. Definition

The simple live interoperability point permits service offerings with formats defined in the first edition of ISO/IEC 23009-1 [4] as well as in DASH-IF IOPs up to version 2. The DASH client ***is not required to parse media segments for proper operation***, but can rely exclusively on the information in the MPD..

4.9.2.2. Service Requirements and Recommendations

Service offerings conforming to this operation shall follow

- The general requirements and guidelines in section 4.3.3
- the MPD Update requirements and guidelines in section 4.4.3
- the requirements and guidelines for service offering of live content in on-demand mode in section 4.6.2
- the synchronization requirements and guidelines in section 4.7.2
- the robustness requirements and guidelines in section 4.8.7

4.9.2.3. Client Requirements and Recommendations

Clients claiming conformance to this operation shall follow

- The general requirements and guidelines in section 4.3.4
- the MPD Update requirements and guidelines in section 4.4.3.5
- the requirements and guidelines for service offering of live content in on-demand mode in section 4.6.3.
- the synchronization requirements and guidelines in section 4.7.3,
- the robustness requirements and guidelines in section 4.8.8,

4.9.3. Main Live Operation

4.9.3.1. Definition

The main live operation permits service offerings with formats defined in the second edition of ISO/IEC 23009-1 [1]. In this case the DASH client ***may be required to parse media segments for proper operation***.

4.9.3.2. Service Requirements and Recommendations

Service offerings claiming conformance to main live shall follow

- the requirements and guidelines in section 4.3.3
- either
    - the requirements and guidelines in section 4.4.3. Note that in this case no profile identifier needs to be added.
- or
    - the segment-based MPD update requirements and guidelines in section 4.5.2. In this case the profile identifier shall be added.
- the requirements and guidelines for service offering of live content in on-demand mode in section 4.6.2
- the synchronization requirements and guidelines in section 4.7.2
- the robustness requirements and guidelines in section 4.8.7

4.9.3.3. Client Requirements and Recommendations

Clients claiming conformance to main live shall follow

- the requirements and guidelines in section 4.3.4,
- the MPD-update requirements and guidelines in section 4.4.3.5,
- the segment-based MPD update requirements and guidelines in section 4.5.3,
- the requirements and guidelines for service offering of live content in on-demand mode in section 4.6.3.

- the synchronization requirements and guidelines in section 4.7.3,
- the robustness requirements and guidelines in section 4.8.8.

5. Ad Insertion in DASH

5.1. Introduction

5.1.1. General

This section provides recommendations for implementing ad insertion in DASH. Specifically, it defines the reference architecture and interoperability points for a DASH-based ad insertion solution.

The baseline reference architecture addresses both server-based and app-based scenarios. The former approach is what is typically used for Apple HLS, while the latter is typically used with Microsoft SmoothStreaming and Adobe HDS.

5.1.2. DASH Concepts

DASH ad insertion relies on several DASH tools defined in ISO/IEC 23009-1 [5], which are introduced in this section. The correspondence between these tools and ad insertion concepts are explained below.

5.1.2.1. Remote Elements

Remote elements are elements that are not fully contained in the MPD document but are referenced in the MPD with an HTTP-URL using a simplified profile of XLink.

A remote element has two attributes, `@xlink:href` and `@xlink:actuate`. `@xlink:href` contains the URL for the complete element, while `@xlink:actuate` specifies the resolution model. The value "`onLoad`" requires immediate resolution at MPD parse time, while "`onRequest`" allows deferred resolution at a time when an XML parser accesses the remote element. In this text we assume deferred resolution of remote elements, unless explicitly stated otherwise. While there is no explicit timing model for earliest time when deferred resolution can occur, the specification strongly suggests it should be close to the expected playout time of the corresponding Period. A reasonable approach is to choose the resolution at the nominal download time of the Segment.

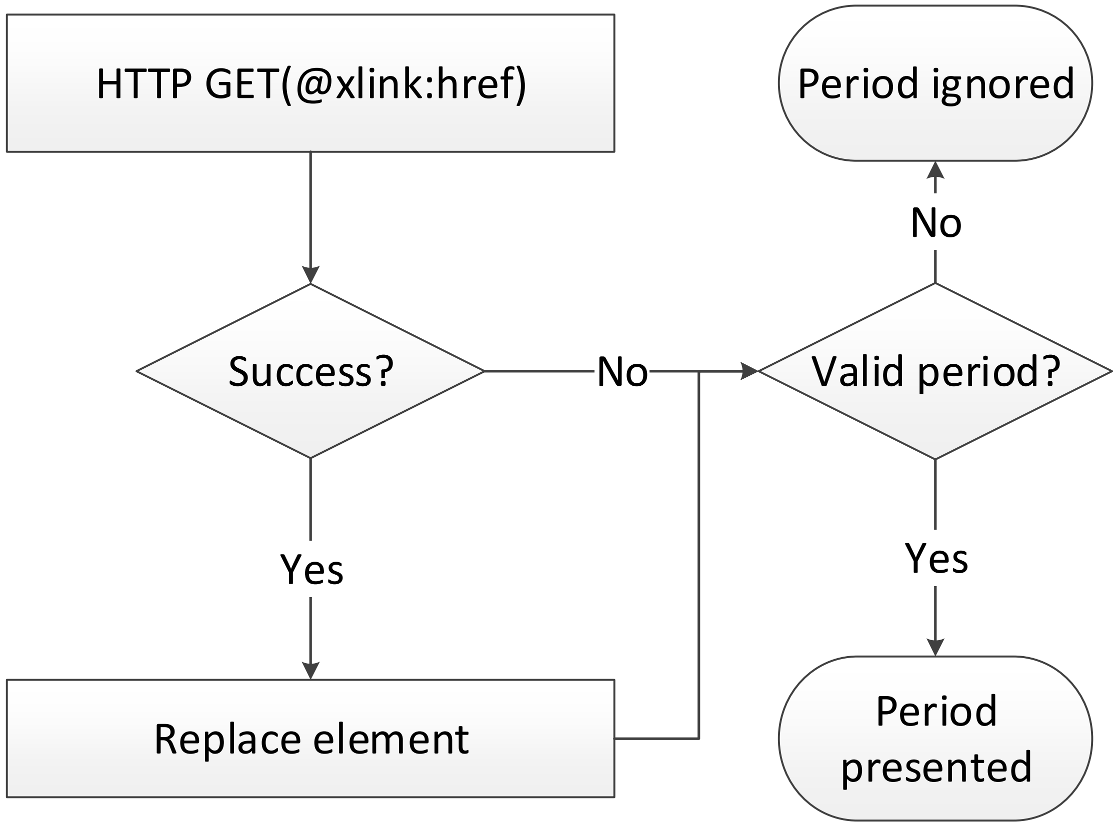


Figure 7: XLink resolution

Resolution (a.k.a. dereferencing) consists of two steps. Firstly, a DASH client issues an HTTP GET request to the URL contained in the `@xlink:href`, attribute of the *in-MPD element*, and the XLink resolver responds with a *remote element entity* in the response content. In case of error response or syntactically invalid remote element entity, the `@xlink:href` and `@xlink:actuate` attributes the client shall remove the *in-MPD element*.

If the value of the `@xlink:href` attribute is `urn:mpeg:dash:resolve-to-zero:2013`, HTTP GET request is not issued, and the in-MPD element shall be removed from the MPD. This special case is used when a remote element can be accessed (and resolved) only once during the time at which a given version of MPD is valid.

If a syntactically valid remote element entity was received, the DASH client will replace in-MPD element with remote period entity.

Once a remote element entity is resolved into a fully specified element, it may contain an `@xlink:href` attribute, which contains a new XLink URL allowing repeated resolution.

Note that the only information passed from the DASH client to the XLink resolver is encoded within the URL. Hence there may be a need to incorporate parameters into it, such as splice time (i.e., *PeriodStart* for the remote period) or cue message.

5.1.2.2. Periods

5.1.2.2.1. Timing

Periods are time-delimited parts of a DASH Media Presentation. The value of *PeriodStart* can be explicitly stated using the **Period@**`start` attribute or indirectly computed using **Period**`@duration` of the previous Periods.

Precise period duration of period *i* is given by *PeriodStart(i+1) – PeriodStart(i)*. This can accommodate the case where media duration of period *i* is slightly longer than the period itself, in which case a client will schedule the start of media presentation for period *i+1* at time *PeriodStart(i+1)*.

Representation`@presentationTimeOffset` specifies the value of the presentation time at *PeriodStart(i)* .

5.1.2.2.2. Segment Availability

In case of dynamic MPDs, Period-level **BaseURL**`@availabilityTimeOffset` allow earlier availability start times. A shorthand notation `@availabilityTimeOffset="INF"` at a Period-level **BaseURL** indicates that the segments within this period are available at least as long as the current MPD is valid. This is the case with stored ad content. Note that DASH also allows specification of `@availabilityTimeOffset` at Adaptation Set and Representation level.

5.1.2.2.3. Seamless transition

The DASH specification says nothing about Period transitions – i.e., there are no guarantees for seamless continuation of playout across the period boundaries. Content conditioning and receiver capability requirements should be defined for applications relying on this functionality. However, Period continuity may be used and signaled using the tools defined in ISO/IEC 23009-1:2014/Amd.3 [5].

5.1.2.2.4. Period labeling

Period-level **AssetIdentifier** descriptors identify the asset to which a given Period belongs. Beyond identification, this can be used for implementation of client functionality that depends on distinguishing between ads and main content (e.g. progress bar and random access).

5.1.2.3. DASH events

DASH events are messages having type, timing and optional payload. They can appear either in MPD (as period-level event stream) or inband, as ISO-BMFF boxes of type `` `emsg` ``. The `` `emsg` `` boxes shall be placed at the very beginning of the Segment, i.e. prior to any media data, so that DASH client needs a minimal amount of parsing to detect them.

DASH defines three events that are processed directly by a DASH client: MPD Validity Expiration, MPD Patch and MPD Update. All signal to the client that the MPD needs to be updated – by providing the publish time of the MPD that should be used, by providing an XML patch that can be applied to the client's in-memory representation of MPD, or by providing a complete new MPD. For details please see section 4.5.

User-defined events are also possible. The DASH client does not deal with them directly – they are passed to an application, or discarded if there is no application willing or registered to process these events. A possible client API would allow an application to register callbacks for specific event types. Such callback will be triggered when the DASH client parses the `emsg` box in a Segment, or when it parses the **Event** element in the MPD.

In the ad insertion context, user-defined events can be used to signal information, such as cue messages (e.g. SCTE 35 [50])

5.1.2.4. MPD Updates

If **MPD**@minimumUpdatePeriod is present, the MPD can be periodically updated. These updates can be *synchronous*, in which case their frequency is limited by **MPD**@minimumUpdatePeriod. In case of the main live profiles MPD updates may be triggered by DASH events. Fir details refer to section 4.5.

When new period containing stored ads is inserted into a linear program, and there is a need to unexpectedly alter this period the inserted media will not carry the `emsg` boxes – these will need to be inserted on-the-fly by proxies. In this case use of synchronous MPD updates may prove simpler.

MPD@publishTime provides versioning functionality: MPD with later publication times include all information that was included all MPDs with earlier publication times.

5.1.2.5. Session information

In order to allow fine-grain targeting and personalization, the identity of the client/viewer, should be known i.e. maintain a notion of a session.

HTTP is a stateless protocol, however state can be preserved by the client and communicated to the server.

The simplest way of achieving this is use of cookies. According to RFC 6265 [37], cookies set via 2xx, 4xx, and 5xx responses must be processed and have explicit timing and security model.

5.1.2.6. Tracking and reporting

The simplest tracking mechanism is server-side logging of HTTP GET requests. Knowing request times and correspondence of segment names to content constitutes an indication that a certain part of the content was requested. If MPDs (or remote element entities) are generated on the fly and identity of the requester is known, it is possible to provide more precise logging. Unfortunately this is a non-trivial operation, as same user may be requesting parts of content from different CDN nodes (or even different CDNs), hence log aggregation and processing will be needed.

Another approach is communicating with existing tracking server infrastructure using existing external standards. An IAB VAST-based implementation is shown in section 5.3.3.6.

DASH Callback events are defined in ISO/IEC 23009-1:2014 AMD3 [5], are a simple native implementation of time-based impression reporting (e.g., quartiles). A callback event is a promise by the DASH client to issue an HTTP GET request to a provided URL at a given offset from *PeriodStart*. The body of HTTP response is ignored.

5.2. Architectures

The possible architectures can be classified based on the location of component that communicates with the ad decision service: a *server-based* approach assumes a generic DASH client and all communication with ad decision services done at the server side (even if this communication is triggered by a client request for a segment, remote element, or an MPD. The *app-based* approach assumes an application running on the end device and controlling one or more generic DASH clients.

Yet another classification dimension is amount of media engines needed for a presentation – i.e., whether parallel decoding needs to be done to allow seamless transition between the main and the inserted content, or content is conditioned well enough to make such transition possible with a single decoder.

Workflows can be roughly classified into *linear* and *elastic*. Linear workflows (e.g., live feed from an event) has ad breaks of known durations which have to be taken: main content will only resume after the end of the break and the programmer / operator needs to fill them with some inserted content. Elastic workflows assume that the duration of an ad break at a given cue location not fixed, thus the effective break length can vary (and can be zero if a break is not taken).

5.3. Server-based Architecture

5.3.1. Introduction

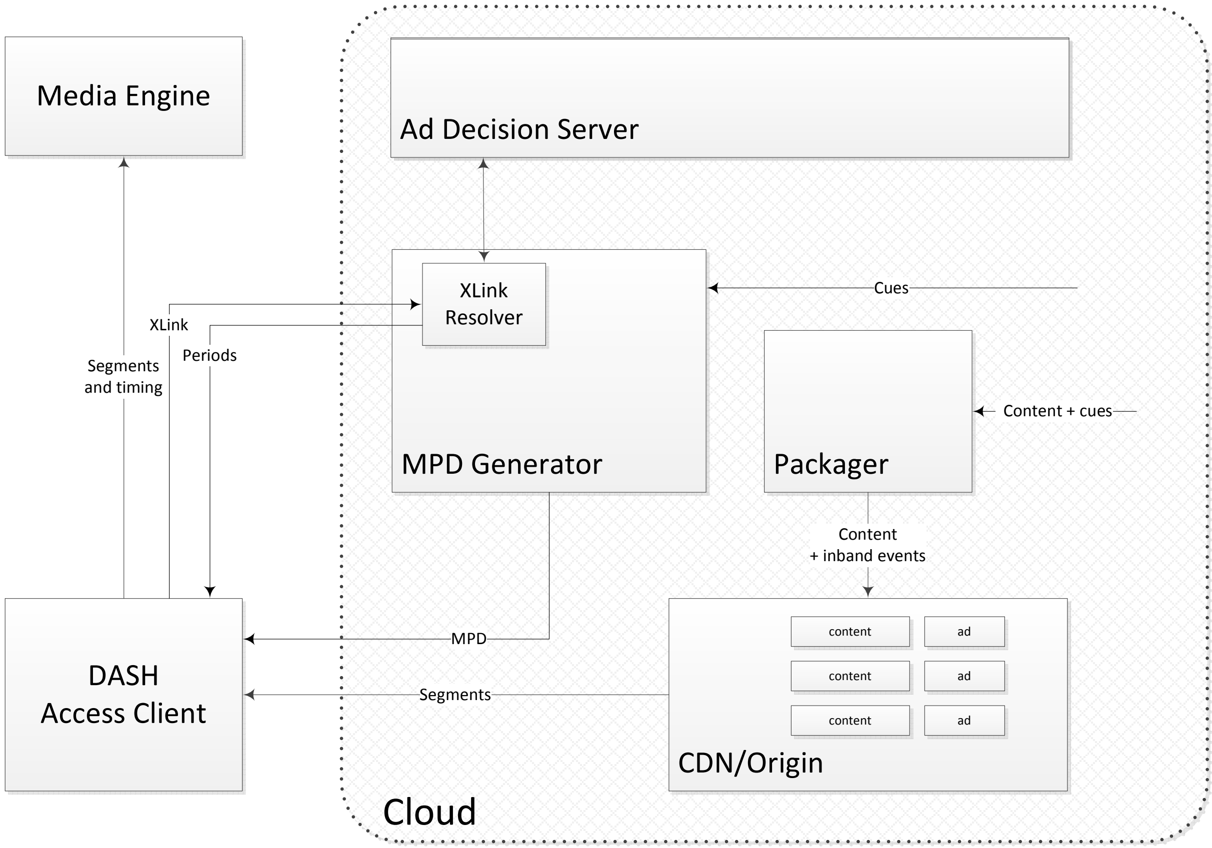


Figure 8: Server-based architecture

In the server-based model, all ad-related information is expressed via MPD and segments, and ad decisions are triggered by client requests for MPDs and for resources described in them (Segments, remote periods).

The server-based model is inherently MPD-centric – all data needed to trigger ad decision is concentrated in the MPD. In case where ad break location (i.e., its start time) is unknown at the MPD generation time, it is necessary to rely on MPD update functionality. The two possible ways of achieving these are described in 5.1.2.4.

In the live case, packager receives feed containing inband cues, such as MPEG-2 TS with SCTE 35 cue messages [50]. The packager ingests content segments into the CDN, passing manifest and cue data to the ad management module. In the on demand case, cues can be provided out of band.

Ad management is located at the server side (i.e., in the cloud), thus all manifest and content conditioning is done at the server side.

5.3.2. Mapping to DASH

5.3.2.1. Period elements

5.3.2.1.1. General

A single ad is expressed as a single **`Period`** element.

Periods with content that is expected to be interrupted as a result of ad insertion should contain explicit start times (**`Period`**`@start`), rather than durations. This allows insertion of new periods without modifying the existing periods. If a period has media duration longer then the distance between the start of this period and the start of next period, use of start times implies that a client will start the playout of the next period at the time stated in the MPD, rather than after finishing the playout of the last segment.

An upcoming ad break is expressed as Period element(s), possibly remote.

5.3.2.1.2. Remote Period elements.

Remote Periods are resolved on demand into one or more than one Period elements. It is possible to embed parameters from the cue message into the XLink URL of the corresponding remote period, in order to have them passed to the ad decision system via XLink resolver at resolution time.

In an elastic workflow, when an ad break is not taken, the remote period will be resolved into a period with zero duration. This period element will contain no adaptation sets.

If a just-in-time remote Period dereferencing is required by use of `@xlink:actuate="onRequest"`, MPD update containing a remote period should be triggered close enough to the intended splice time. This can be achieved using MPD Validity events and full-fledged MPD update, or using MPD Patch and MPD Update events (see sec. 5.1.2.4 and 5.1.2.3). However, due to security reasons MPD Patch and MPD Update events should only be used with great care.

In case of **`Period`**`@xlink:actuate="onRequest"`, MPD update and XLink resolution should be done sufficiently early to ensure that there are no artefacts due to insufficient time given to download the inserted content. Care needs to be taken so that the client is given a sufficient amount of time to (a) request and receive MPD update, and (b) dereference the upcoming remote period.

> NOTE: It may be operationally simpler to avoid use of **`Period`**`@xlink:actuate="onRequest"`, dereferencing in case of live content.

5.3.2.2. Asset Identifiers

`AssetIdentifier` descriptors identify the asset to which a Period belongs. This can be used for implementation of client functionality that depends on distinguishing between ads and main content (e.g. progress bar).

Periods with same **`AssetIdentifier`** should have identical Adaptation Sets, Initialization Segments and same DRM information (i.e., DRM systems, licenses). This allows reuse of at least some initialization data across periods of the same asset, and ensures seamless continuation of playback if inserted periods have zero duration. Period continuity may be signaled.

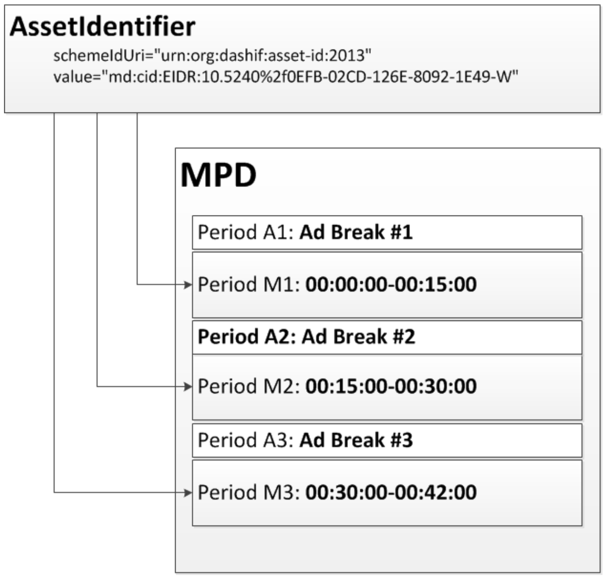


Figure 9: Using an asset identifier

5.3.2.3. MPD updates

MPD updates are used to implement dynamic behavior. An updated MPD may have additional (possibly – remote) periods. Hence, MPD update should be triggered by the arrival of the first cue message for an upcoming ad break. Ad breaks can also be canceled prior to their start, and such cancellation will also trigger an MPD update.

Frequent regular MPD updates are sufficient for implementing dynamic ad insertion. Unfortunately they create an overhead of unnecessary MPD traffic – ad breaks are rare events, while MPD updates need to be frequent enough if a cue message is expected to arrive only several seconds before the splice point. Use of HTTP conditional GET requests (i.e., allowing the server to respond with "304 Not Modified" if MPD is unchanged) is helpful in reducing this overhead, but asynchronous MPD updates avoid this overhead entirely.

DASH events with scheme `"urn:mpeg:dash:event:2013"` are used to trigger asynchronous MPD updates.

The simple mapping of live inband cues in live content into DASH events is translating a single cue into an MPD Validity expiration event (which will cause an MPD update prior to the splice time). MPD Validity expiration events need to be sent early enough to allow the client request a new MPD, resolve XLink (which may entail communication between the resolver and ADS), and, finally, download the first segment of the upcoming ad in time to prevent disruption of service at the splice point.

If several \``emsg`\` boxes are present in a segment and one of them is the MPD Validity Expiration event, \``emsg`\` carrying it shall always appear first.

5.3.2.4. MPD events

In addition to tracking events (ad starts, quartile tracking, etc.) the server may also need to signal additional metadata to the video application. For example, an ad unit may contain not only inline linear ad content (that is to be played before, during, or after the main presentation), it may also contain a companion display ad that is to be shown at the same time as the video ad. It is important that the server be able to signal both the presence of the companion ad and the additional tracking and click-through metadata associated with the companion.

With that said, there is no need to have a generic DASH client implement this functionality – it is enough to provide opaque information that the client would pass to an external module. Event `@schemeIdUri` provides us with such addressing functionality, while MPD events allow us to put opaque payloads into the MPD.

5.3.3. Workflows

In the workflows below we assume that our inputs are MPEG-2 transport streams with embedded SCTE 35 cue messages [50]. In our opinion this will be a frequently encountered deployment, however any other in-band or out-of-band method of getting cue messages and any other input format lend themselves into the same model.

5.3.3.1. Linear

A real-time MPEG-2 TS feed arrives at both packager and MPD generator. While real-time multicast feeds are a very frequently encountered case, the same workflow can apply to cases such as ad replacement in a pre-recorded content (e.g., in time-shifting or PVR scenarios).

MPD generator generates dynamic MPDs. Packager creates DASH segments out of the arriving feed and writes them into the origin server. Client periodically requests the MPDs so that it has enough time to transition seamlessly into the ad period.

Packager and MPD generator may be tightly coupled (e.g. co-located on the same physical machine), or loosely coupled as they both are synchronized only to the clock of the feed.

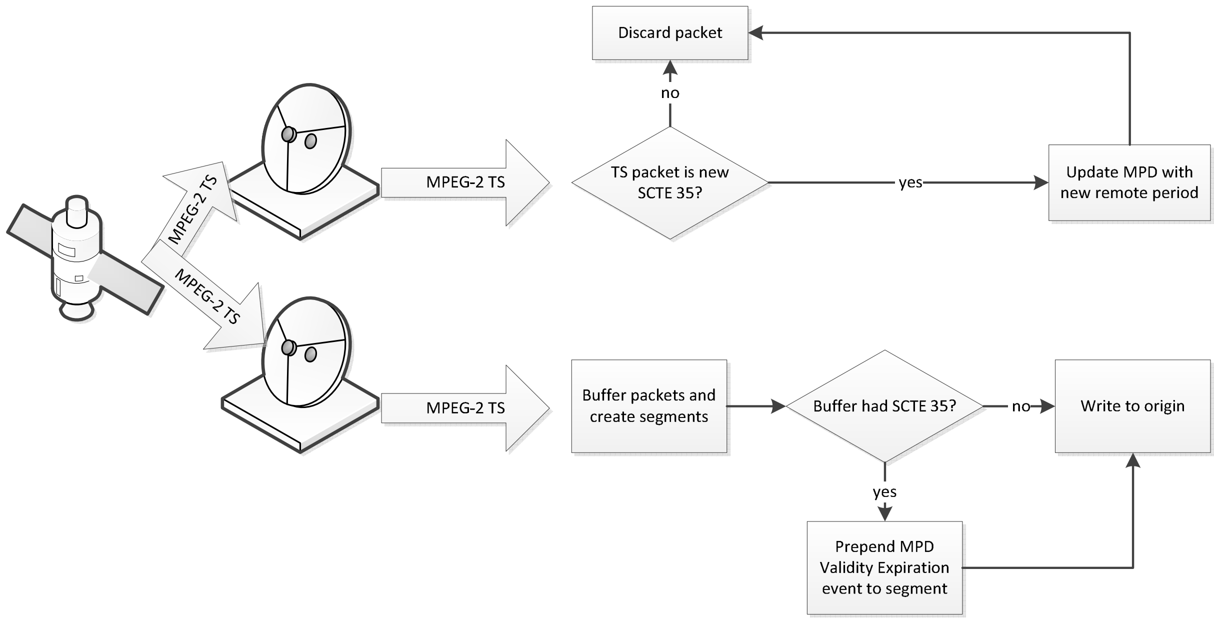


Figure 10: Live Workflow

5.3.3.1.1. Cue Interpretation by the MPD generator

When an SCTE 35 cue message indicating an upcoming splice point is encountered by the MPD generator, the latter creates a new MPD for the same program, adding a remote period to it.

The **Period**@start attribute of the inserted period has splice_time() translated into the presentation timeline. Parameters derived from the cue message are inserted into the **Period**@xlink:href attribute of the inserted period. Examples below show architectures that allow finer targeting.

5.3.3.1.1.1. Example 1: Immediate ad decision

MPD generator keeps an up-to-date template of an MPD. At each cue message arrival, the generator updates its template. At each MPD request, the generator customizes the request based on the information known to it about the requesting client. The generator contacts ad decision server and produces one or more non-remote ad periods. In this case XLink is not needed.

5.3.3.1.1.2. Example 2: Stateful cue translation

MPD generator keeps an up-to-date template of an MPD. At each cue message arrival, the generator updates its template. At each MPD request, the generator customizes the request based on the information known to it about the requesting client.

The operator targets separately male and female audiences. Hence, the generator derives this from the information it has regarding the requesting client (see 5.1.2.5), and inserts an XLink URL with the query parameter ?gender=male for male viewers, and ?gender=female for the female viewers.

Note that this example also showcases poor privacy practices – would such approach be implemented, both parameter name and value should be encrypted or TLS-based communication should be used

5.3.3.1.1.3. Example 3: Stateless cue translation

At cue message arrival, the MPD generator extracts the entire SCTE 35 splice_info_section (starting at the table_id and ending with the CRC_32) into a buffer. The buffer is then encoded into URL-safe base64url format according to RFC 4648 [56], and inserted into the XLink URL of a new remote Period element. splice_time is translated into **Period**@start attribute. The new MPD is pushed to the origin.

Note: this example is a straightforward port of the technique defined for SCTE 67 [51], but uses base64url and not base64 encoding as the section is included in a URI.

5.3.3.1.2. Cue Interpretation by the packager

Cue interpretation by the packager is optional and is an optimization, rather than core functionality.

On reception of an SCTE 35 cue message signaling an upcoming splice, an `emsg` with MPD Validity Expiration event is inserted into the first available segment. This event triggers an MPD update, and not an ad decision, hence the sum of the earliest presentation time of the `emsg`-bearing segment and the `emsg`.presentation_time_delta should be sufficiently earlier than the splice time. This provides the client with sufficient time to both fetch the MPD and resolve XLink.

splice_time() of the cue message is translated into the media timeline, and last segment before the splice point is identified. If needed, the packager can also finish the segment at the splice point and thus having a segment shorter than its target duration.

5.3.3.1.3. Multiple cue messages

There is a practice of sending several SCTE 35 cue messages for the same splice point (e.g., the first message announces a splice in 6 seconds, the second arrives 2 seconds later and warns about the same splice in 4 seconds, etc.). Both the packager and the MPD generator react on the same first message (the 6-sec warning in the example above), and do nothing about the following messages.

5.3.3.1.4. Cancelation

It is possible that the upcoming (and announced) insertion will be canceled (e.g., ad break needed to be postponed due to overtime). Cancelation is announced in a SCTE 35 cue message.

When cancelation is announced, the packager will insert the corresponding `emsg` event and the MPD generator will create a newer version of the MPD that does not contain the inserted period or sets its duration to zero. This implementation maintains a simpler less-coupled server side system at the price of an increase in traffic.

5.3.3.1.5. Early termination

It is also possible that a planned ad break will need to be cut short – e.g., an ad will be cut short and there will be a switch to breaking news. The DASH translation of this would be creating an `emsg` at the packager and updating the MPD appropriately. Treatment of early termination here would be same as treatment of a switch from main content to an ad break.

It is easier to manipulate durations when **Period**@duration is absent and only **Period**@start is used – this way attributes already known to the DASH client don't change.

5.3.3.1.6. Informational cue messages

SCTE 35 can be used for purposes unrelated to signaling of placement opportunities. Examples of such use are content identification and time-of-day signaling. Triggering MPD validity expiration and possibly XLink resolution in this case may be an overreaction.

5.3.3.1.7. Ad decision

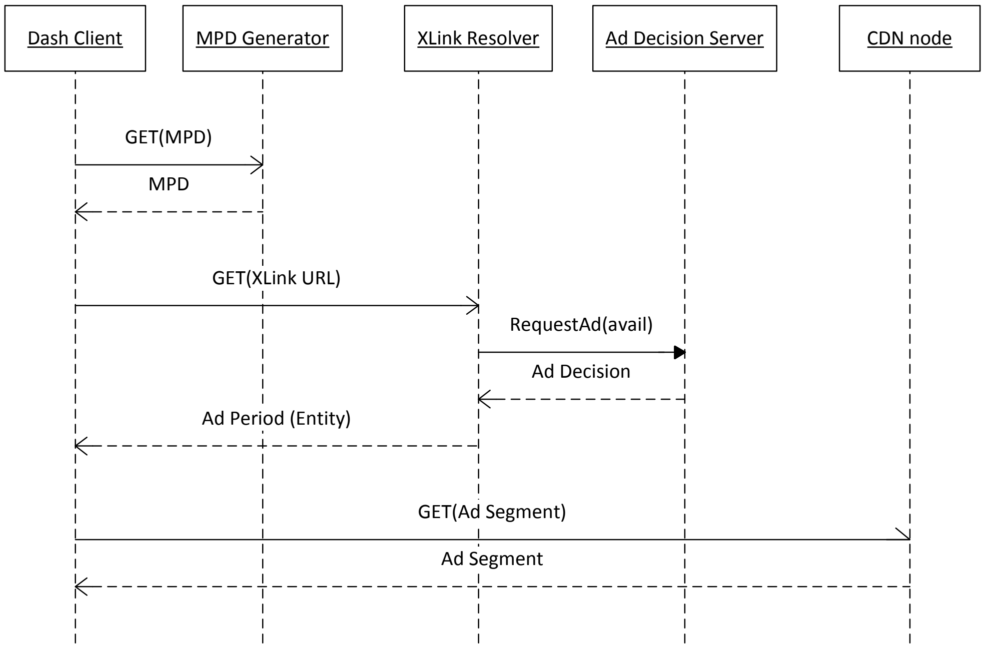


Figure 11: Ad Decision

A client will attempt to dereference a remote period element by issuing an HTTP GET for the URL that appears in **Period**@xlink:href. The HTTP server responding to this request (XLink resolver) will contact the ad decision service, possibly passing it parameters known from the request URL and from client information available to it from the connection context. In case described in 5.3.3.1.1.3, the XLink resolver has access to a complete SCTE 35 message that triggered the splice.

The ad decision service response identifies the content that needs to be presented, and given this information the XLink resolver can generate one or more Period elements that would be then returned to the requesting DASH client.

A possible optimization is that resolved periods are cached – e.g. in case of 5.3.3.1.1.1 "male" and "female" versions of the content are only generated once in *T* seconds, with HTTP caching used to expire the cached periods after *T* seconds.

5.3.3.2. On Demand

In a VoD scenario, cue locations are known ahead of time. They may be available multiplexed into the mezzanine file as SCTE 35 or SCTE 104, or may be provided via an out-of-band EDL.

In VoD workflows both cue locations and break durations are known, hence there is no need for a dynamic MPD. Thus cue interpretation (which is same as in 5.3.3.1) can occur only once and result in a static MPD that contains all remote elements with all Period elements having **Period**@start attribute present in the MPD.

In elastic workflows ad durations are unknown, thus despite our knowledge of cue locations within the main content it is impossible to build a complete presentation timeline. **Period**@duration needs to be used. Remote periods should be dereferenced only when needed for playout. In case of a “jump” – random access into an arbitrary point in the asset – it is a better practice not to dereference Period elements when it is possible to determine the period from which the playout starts using **Period**@duration and asset identifiers. The functionality described in 5.3.3.1 is sufficient to address on-demand cases, with the only difference that a client should be able to handle zero-duration periods that are a result of avails that are not taken.

5.3.3.3. Capture to VoD

Capture to VoD use case is a hybrid between pure linear and on demand scenarios: linear content is recorded as it is broadcast, and is then accessible on demand. A typical requirement is to have the content available with the original ad for some time, after which ads can be replaced

There are two possible ways of implementing the capture-to-VoD workflow.

The simplest is treating capture-to-VoD content as plain VoD, and having the replacement policy implemented on the XLink resolver side. This way the same Period element(s) will be always returned to the same requester within the window where ad replacement is disallowed; while after this window the behavior will be same as for any on-demand content. An alternative implementation is described in 5.3.3.4 below.

5.3.3.4. Slates and ad replacement

A content provider (e.g., OTT) provides content with ad breaks filled with its own ads. An ISP is allowed to replace some of these with their own ads. Conceptually there is content with slates in place of ads, but all slates can be shown and only some can be replaced.

An ad break with a slate can be implemented as a valid in-MPD Period element that also has XLink attributes. If a slate is replaceable, XLink resolution will result in new Period element(s), if not – the slate is played out.

5.3.3.5. Blackouts and Alternative content

In many cases broadcast content cannot be shown to a part of the audience due to contractual limitations (e.g., viewers located close to an MLB game will not be allowed to watch it, and will be shown some alternative content). While unrelated to ad insertion per se, this use case can be solved using the same “default content” approach, where the in-MPD content is the game and the alternative content will be returned by the XLink resolver if the latter determines (in some unspecified way) that the requester is in the blackout zone.

5.3.3.6. Tracking and reporting

A Period, either local or a remote entity, may contain an EventStream element with an event containing IAB VAST 3.0 Ad element [49]. DASH client does not need to parse the information and act accordingly – if there is a listener to events of this type, this listener can use the VAST 3.0 Ad element to implement reporting, tracking and companion ads. The processing done by this listener does not have any influence on the DASH client, and same content would be presented to both “vanilla” DASH client and the player in which a VAST module registers with a DASH client a listener to the VAST 3.0 events.

An alternative implementation uses DASH Callback events. While DASH specification permits both inband and MPD Callback events, inband callback events shall not be used.

5.3.4. Examples

5.3.4.1. MPD with mid-roll ad breaks and default content

In this example, a movie ("Top Gun") is shown on a linear channel and has two mid-roll ad breaks. Both breaks have default content that will be played if the XLink resolver chooses not to return new Period element(s) or fails.

In case of the first ad break, SCTE 35 cue message is passed completely to the XLink resolver, together with the corresponding presentation time.

In case of the second ad break, proprietary parameters u and z describe the main content and the publishing site.

```
<?xml version="1.0"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="urn:mpeg:dash:schema:mpd:2011"
    xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 DASH-MPD.xsd"
    type="dynamic"
    minimumUpdatePeriod="PT2S"
    timeShiftBufferDepth="PT600S"
    minBufferTime="PT2S"
    profiles="urn:mpeg:dash:profile:isoff-live:2011"
    availabilityStartTime="2012-12-25T15:17:50">
    <BaseURL>http://cdn1.example.com/</BaseURL>
    <BaseURL>http://cdn2.example.com/</BaseURL>

    <!-- Movie -->
    <Period start="PT0.00S" duration="PT600.6S" id="movie period #1">
        <AssetIdentifier schemeIdUri="urn:org:dashif:asset-id:2013"
            value="md:cid:EIDR:10.5240%2f0EFB-02CD-126E-8092-1E49-W">
        <AdaptationSet mimeType="video/mp4" codecs="avc1.640828"
            frameRate="24000/1001" segmentAlignment="true" startWithSAP="1">
            <BaseURL>video_1/</BaseURL>
            <SegmentTemplate timescale="90000" initialization="$Band-
width%/init.mp4v"
                media="$Bandwidth$/$Number%05d$.mp4v"/>
            <Representation id="v0" width="320" height="240" bandwidth="250000"/>
            <Representation id="v1" width="640" height="480" bandwidth="500000"/>
            <Representation id="v2" width="960" height="720" bandwidth="1000000"/>
        </AdaptationSet>
    </Period>

    <!-- Mid-roll advertisement, passing base64url-coded SCTE 35 to XLink resolver -
->
    <Period duration="PT60.6S" id="ad break #1"
            xlink:href="https://adserv.com/avail.mpd?time=54054000&id=1234567&
                 cue=DAIAAAAAAAAAAAQAAZ_I0VniQAQAgBDVUVJQAAAAH+cAAAAAA=="
            xlink:actuate="onRequest" >

        <!-- Default content, replaced by elements from remote entity -->
        <AdaptationSet mimeType="video/mp4" codecs="avc1.640828"
                      frameRate="30000/1001"
            segmentAlignment="true" startWithSAP="1">
            <BaseURL availabilityTimeOffset="INF">default_ad/</BaseURL>
```

```
            <SegmentTemplate timescale="90000" initialization="$Band-
width%/init.mp4v"
                media="$Bandwidth%/$Time$.mp4v"/>
            <Representation id="v0" width="320" height="240" bandwidth="250000"/>
            <Representation id="v1" width="640" height="480" bandwidth="500000"/>
            <Representation id="v2" width="960" height="720" bandwidth="1000000"/>
        </AdaptationSet>
    </Period>

    <!–Movie, cont'd -->
    <Period duration="PT600.6S" id="movie period #2">
        <AssetIdentifier schemeIdUri="urn:org:dashif:asset-id:2013"
            value="md:cid:EIDR:10.5240%2f0EFB-02CD-126E-8092-1E49-W">
        <AdaptationSet mimeType="video/mp4" codecs="avc1.640828"
                       frameRate="24000/1001"
            segmentAlignment="true" startWithSAP="1">
            <BaseURL>video_2/</BaseURL>
            <SegmentTemplate timescale="90000" initialization="$Band-
width%/init.mp4v"
                media="$Bandwidth%/$Time$.mp4v"/>
            <Representation id="v0" width="320" height="240" bandwidth="250000"/>
            <Representation id="v1" width="640" height="480" bandwidth="500000"/>
            <Representation id="v2" width="960" height="720" bandwidth="1000000"/>
        </AdaptationSet>
    </Period>

    <!-- Mid-roll advertisement, using proprietary parameters  -->
    <Period start="PT60.6S" id="ad break #2"
        xlink:href=”https://adserv.com/avail.mpd?u=0EFB-02CD-126E-8092-1E49-
W&z=spam”
        xlink:actuate="onRequest" >

        <!-- Default content, replaced by elements from remote entity -->
        <AdaptationSet mimeType="video/mp4" codecs="avc1.640828"
                       frameRate="30000/1001"
            segmentAlignment="true" startWithSAP="1">
            <BaseURL availabilityTimeOffset="INF">default_ad2/</BaseURL>
            <SegmentTemplate timescale="90000" initialization="$Band-
width%/init.mp4v"
                media="$Bandwidth%/$Time$.mp4v"/>
            <Representation id="v0" width="320" height="240" bandwidth="250000"/>
            <Representation id="v1" width="640" height="480" bandwidth="500000"/>
            <Representation id="v2" width="960" height="720" bandwidth="1000000"/>
        </AdaptationSet>
    </Period>
</MPD>
```

Figure 12: Example of MPD for "Top Gun" movie

5.4. App-based Architecture

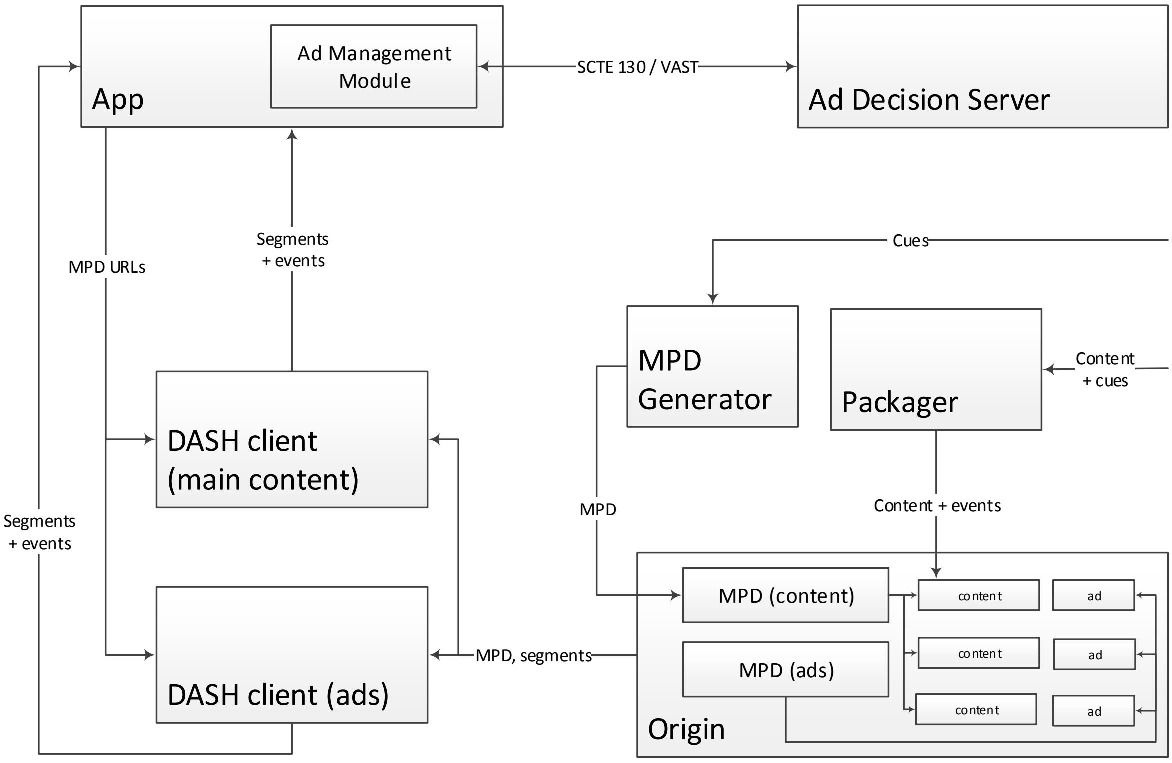


Figure 13: App-based architecture

Inputs in this use case are same as the ones described in sec. 5.3. At the packaging stage, cues are translated into a format readable by the app or/and DASH client and are embedded into media segments or/and into the manifest

Ad management module is located at the client side. The DASH client receives manifest and segments, with cues embedded in either one of them or in both.

Cue data is passed to the ad management module, which contacts the ad decision service and receives information on content to be played. This results in an MPD for an inserted content and a splice time at which presentation of main content is paused and presentation of the inserted content starts.

Note that this architecture does not assume multiple decoders – with careful conditioning it is possible to do traditional splicing where inserted content is passed to the same decoder. In this case it is necessary to keep a player state and be able to initialize a player into this state.

5.4.1. Mapping to DASH

This section details mapping of elements of the reference architecture into DASH concepts per the 2nd edition of the specification (i.e., ISO/IEC 23009-1:2014).

5.4.1.1. MPD

Each ad decision results in a separate MPD. A single MPD contains either main content or inserted content; existence of multiple periods or/and remote periods is possible but not essential.

5.4.1.2. SCTE 35 events

5.4.1.2.1. General

Cue messages are mapped into DASH events, using inband `emsg` boxes and/or in-MPD events. Note that SCTE 35 cue message may not be sufficient by itself.

The examples below show use of SCTE 35 in user-defined events, and presentation time indicates the timing in within the Period.

Figure 14 below shows the content of an `emsg` box at the beginning of a segment with earliest presentation time *T*. There is a 6-sec warning of an upcoming splice – delta to splice time is indicated as 6 seconds – and duration is given as 1 minute. This means that an ad will start playing at time *T + 6* till *T + 66*. This example follows a practice defined in SCTE DVS 1208.

scheme_id_uri="urn:scte:scte35:2013:xml"
value=1001
timescale=90000
presentation_time_delta=540000
duration=5400000
id=0
message_data[]=

```
<SpliceInfoSection ptsAdjustment="0" scte35:tier="22">
   <SpliceInsert
       spliceEventId="111" spliceEventCancelIndicator="false"
       outOfNetworkIndicator="true" uniqueProgramId="65535"
       availNum="1" availsExpected="2" spliceImmediateFlag="false">
       <Program><SpliceTime ptsTime="122342"/></Program>
       <BreakDuration autoReturn="false" duration="5400000"/>
   </SpliceInsert>
   <AvailDescriptor scte35:providerAvailId="332"/>
</SpliceInfoSection>
```

Figure 14 Inband carriage of SCTE 35 cue message

Figure 15 below shows the same example with an in-MPD SCTE35 cue message. The difference is in the in-MPD event the splice time is relative to the Period start, rather than to the start of the event-carrying segment. This figure shows a one-minute ad break 10 minutes into the period.

```
<EventStream schemeIdUri="urn:scte:scte35:2013:xml">
    <Event timescale="90000" presentationTime="54054000" duration="5400000" id="1">
        <scte35:SpliceInfoSection scte35:ptsAdjustment="0" scte35:tier="22">
             <scte35:SpliceInsert
                  scte35:spliceEventId="111"
                  scte35:spliceEventCancelIndicator="false"
                  scte35:outOfNetworkIndicator="true"
                  scte35:uniqueProgramId="65535"
                  scte35:availNum="1"
                  scte35:availsExpected="2"
                  scte35:spliceImmediateFlag="false">
                  <scte35:Program>

<!-- Event timing is given by Event@presentationTime,  -->

<!-- splice_time() processing is up to the application -->
                      <scte35:SpliceTime scte35:ptsTime="122342"/>
                  </scte35:Program>
                  <scte35:BreakDuration

scte35:autoReturn="false" scte35:duration="5400000"/>
             </scte35:SpliceInsert>
             <scte35:AvailDescriptor scte35:providerAvailId="332"/>
         </scte35:SpliceInfoSection>
    </Event>
</EventStream>
```

Figure 15: In-MPD carriage of SCTE 35 cue message

Note: for brevity purposes SCTE 35 2014 allows use of base64-encoded section in **`Signal.Binary`** element as an alternative to carriage of a completely parsed cue message.

Normative definitions of carriage of SCTE 35 cue messages are in SCTE DVS 1202 sec 6.8.4 (MPD) and SCTE DVS 1208 sec 8.3.3.

5.4.1.3. Asset Identifiers

See sec. 5.3.2.2 for details.

5.4.2. Workflows

5.4.2.1. Linear

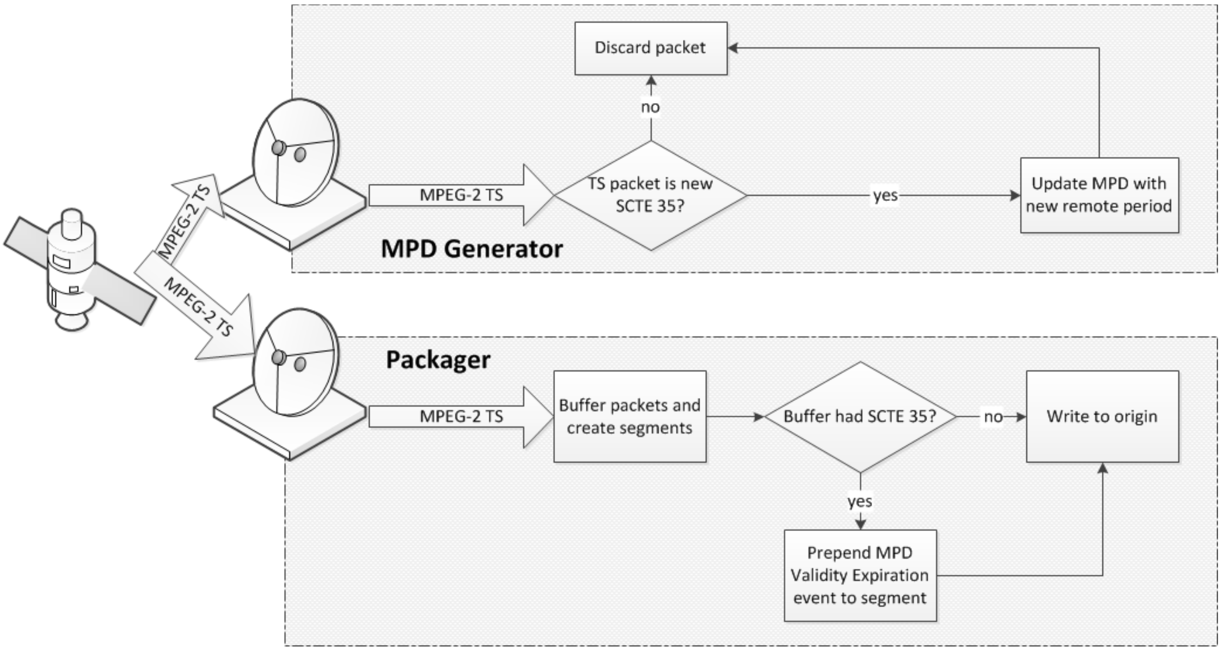


Figure 16: Linear workflow for app-driven architecture

A real-time MPEG-2 TS feed arrives at a packager. While real-time multicast feeds are a very frequently encountered case, the same workflow can apply to cases such as ad replacement in a pre-recorded content (e.g., in time-shifting or PVR scenarios).

Packager creates DASH segments out of the arriving feed and writes them into the origin server. The packager translates SCTE 35 cue messages into inband DASH events, which are inserted into media segments.

MPD generator is unaware of ad insertion functionality and the packager does the translation of SCTE 35 cue messages into inband user-defined DASH events. On reception of an SCTE 35 cue message signaling an upcoming splice, a `` `emsg` `` with a translation of the cue message in its `` `emsg`.message_data[] `` field is inserted into the most recent Segment. This event triggers client interaction with an ad decision server, hence the sum of the earliest presentation time of the `` `emsg` ``-bearing segment and the `` `emsg`.presentation_time_delta `` should be a translation of `splice_time()` into the media timeline.

An alternative implementation which is more compatible with server-based architecture in section 5.3, an MPD generator can generate separate MPDs for both server-based and app-based architectures creating remote periods for server-based and in-MPD SCTE 35 events for app-based architectures, while a packager can insert inband MPD validity expiration events.

A DASH client will pass the event to the app controlling it (e.g., via a callback registered by the app). The app will interpret the event and communicate with the ad decision server using some interface (e.g., VAST). This interface is out of the scope of this document.

The communication with ad decision service will result in an MPD URL. An app will pause the presentation of the main content and start presentation of the inserted content. After presenting the inserted content the client will resume presentation of the main content. This assumes either proper conditioning of the main and inserted content or existence of separate client and decoder for inserted content. The way pause/resume is implemented is internal to the API of the DASH client. Interoperability may be achieved by using the DASH MPD fragment interface, see ISO/IEC 23009-1 [4], Annex C.4.

5.4.2.2. On Demand

As in the server-based case, functionality defined for the live case is sufficient. Moreover, the fact that that app-based implementation relies heavily on app's ability to pause and resume the DASH client, support for elastic workflows is provided out of the box.

In the on demand case, as cue locations are well-known, it is advantageous to provide a static MPD with SCTE 35 events than run a dynamic service that relies on inband events.

5.5. Extensions for ad insertion

5.5.1. Asset Identifiers

`AssetIdentifier` descriptor shall be used for distinguishing parts of the same asset within a multi-period MPD, hence it shall be used for main content.

In order to enable better tracking and reporting, unique IDs should be used for different assets.

In the absence of other asset identifier schemes, a DASH-IF defined scheme may be used with the value of `@schemeIdUri` set to `"urn:org:dashif:asset-id:2014"`. If used, the value of `@value` attribute descriptor shall be a MovieLabs ContentID URN ([54], 2.2.1) for the content. It shall be the same for all parts of an asset. Preferred schemes are EIDR (main content) and Ad-ID (advertising).

If a Period has one-off semantics (i.e., an asset is completely contained in a single period, and its continuation is not expected in the future), the author shall not use asset identifier on these assets.

Periods that do not contain non-remote **`AdaptationSet`** elements, as well as zero-length periods shall not contain the **`AssetIdentifier`** descriptor.

5.5.2. Remote Periods

An MPD may contain remote periods, some of which may have default content. Some of which are resolved into multiple Period elements.

After dereferencing MPD may contain zero-length periods or/and remote Periods.

In case of **`Period`**`@xlink:actuate="onRequest"`, MPD update and XLink resolution should be done sufficiently early to ensure that there are no artefacts due to insufficient time given to download the inserted content.

5.5.3. User-defined events

5.5.3.1. Cue message

Cue messages used in app-driven architecture shall be SCTE 35 events [50]. SCTE 35 event carriage is defined in SCTE DVS 1202 (MPD) and SCTE DVS 1208 (inband). For MPD events, the XML schema is defined in SCTE 35 2014 [50] and allows either XML representation or concise base64-coded representation.

NOTE: PTS offset appearing in SCTE 35 shall be ignored, and only DASH event timing mechanism may be used to determine splice points.

5.5.3.2. Reporting

MPD events with embedded IAB VAST 3.0 [49] response may be used for reporting purposes.

If only time-based reporting is required (e.g., reporting at start, completion, and quartiles), use of DASH callback event may be a simpler native way of implementing tracking. Callback events are defined in ISO/IEC 23009-1:2014 AMD3 [5].

5.5.3.3. Ad Insertion Event Streams

Recommended Event Stream schemes along with their scheme identifier for app-driven ad insertion are:

1. `"urn:scte:scte35:2013:bin"` for inband SCTE 35 events containing a complete SCTE 35 section in binary form, as defined in SCTE DVS 1208.
2. `“urn:scte:scte35:2014:xml+bin”` for MPD SCTE 35 events containing only base64 cue message representation, as defined in SCTE DVS 1202.

   NOTE: the content of Event element is an XML representation of the complete SCTE 35 cue message, that contains **`Signal.Binary`** element rather than the **`Signal.SpliceInfoSection`** element, both defined in SCTE 35 2014.
3. `"urn:dashif:org:vast30:2014"` for VAST3.0 [49].

5.6. Interoperability Aspects

5.6.1. Server-based Ad insertion

For server-based ad insertion, the following aspects needs to be taken into account:

- Service offerings claiming conformance to server-based ad insertion shall follow the requirements and guidelines for service offerings in sections 5.3.2, 5.5.1, and 5.5.2..
- Clients claiming conformance to server-based ad insertion shall follow shall follow the requirements and guidelines for clients in section 5.3.2, 5.5.1, and 5.5.2. .

5.6.2. App-based Ad Insertion

For app-based ad insertion, the logic for ad insertion is outside the scope of the DASH client. The tools defined in section 5.4 and 5.5 may be used to create an interoperable system that includes DASH-based delivery and ad insertion logic.

6. Media Coding Technologies

6.1. Introduction

In addition to DASH-specific constraints, DASH-IF IOPs also adds restrictions on media codecs and other technologies. This section provides an overview on technologies for different media components and how they fit into the DASH-related aspects of DASH-IF IOPs.

6.2. Video

6.2.1. General

The codec considered for basic video support up to 1280 x 720p at 30 fps is H.264 (AVC) Progressive High Profile Level 3.1 decoder [9]. This choice is based on the tradeoff between content availability, support in existing devices and compression efficiency.

Further, it is recognized that certain clients may only be capable to operate with H.264/AVC "Progressive" Main Profile Level 3.0 and therefore content authors may provide and signal a specific subset of DASH-AVC/264.

Notes

- H.264 (AVC) Progressive High Profile Level 3.1 decoder [9] can also decode any content that conforms to
    - H.264 (AVC) Constrained Baseline Profile up to Level 3.1
    - H.264 (AVC) "Progressive" Main Profile up to Level 3.1.
- H.264 (AVC) H.264/AVC "Progressive" Main Profile Level 3.0 decoder [9] can also decode any content that conforms to H.264 (AVC) Constrained Baseline Profile up to Level 3.0.

Further, the choice for HD extensions up to 1920 x 1080p and 30 fps is H.264 (AVC) Progressive High Profile Level 4.0 decoder [9].

The High Efficiency Video Coding (HEVC) resulted from a joint video coding standardization project of the ITU-T Video Coding Experts Group (ITU-T Q.6/SG 16) and ISO/IEC Moving Picture Experts Group (ISO/IEC JTC 1/SC 29/WG 11). The final specification is available here [20]. Additional background information may be found at http://hevc.info.

The DASH-IF is interested in providing Interoperability Points and Extensions for established codec configurations. It is not the intent of the DASH-IF to define typically deployed HEVC profiles/levels or the associated source formats. However, at the same time it is considered to provide implementation guidelines supported by test material for DASH-based delivery as soon as the industry has converged to profile/level combinations in order to support a dedicated format. For this version of this document the following is considered:

- For HEVC-based video, it is expected that the minimum supported format is 720p. The codec considered to support up to 1280 x 720p at 30 fps is HEVC Main Profile Main Tier Level 3.1 [20].

- The choice for 8-bit HD extensions based on HEVC to support up to 2048 x 1080 and 60 fps is HEVC Main Profile Main Tier Level 4.1 [20].
- The choice for 10-bit HD extensions based on HEVC to support up to 2048 x 1080 and 60 fps and 10 bit frame depth is HEVC Main10 Profile Main Tier Level 4.1 [20].

Other profile/level combinations will be considered in updated versions of this document.

6.2.2. DASH-specific aspects for H.264/AVC video

For the integration of the above-referred codecs in the context of DASH, the following applies for H.264 (AVC):

- The encapsulation of H.264/MPEG-4 AVC video data is based on the ISO BMFF as defined in ISO/IEC 14496-15 [10].
- Clients shall to support H.264/AVC sample entries when SPS/PPS is provided in the Initialization Segment only according to ISO/IEC 14496-15, [10], i.e. sample entry '`avc1`'.
- Clients shall support Inband Storage for SPS/PPS based ISO/IEC 14496-15, [10], i.e. sample entry '`avc3`'.
- Service offerings using H.264/AVC may use sample entry '`avc1`' or '`avc3`'.
- SAP types 1 and 2 correspond to IDR-frames in [9].
- The signaling of the different video codec profile and levels for the codecs parameters according to RFC6381 [11] is documented in Table 14. Note that any of the codecs present in Table 2 conforms to the profile level combination that is supported in DASH-AVC/264.
- Additional constraints within one Adaptation Set are provided in section 6.2.5.

  Note: For a detailed description on how to derive the signaling for the codec profile for H.264/AVC, please refer to DVB DASH, section 5.1.3.

Table 14 H.264 (AVC) Codecs parameter according to RFC6381 [11]

Profile	Level	Codec Parameter
H.264 (AVC) "Progressive" Main Profile	3.0	`avc[1,3].4DY01E`
H.264 (AVC) Progressive High Profile	3.1	`avc[1,3].64Y01F`
	4.0	`avc[1,3].64Y028`

6.2.3. DASH-specific aspects for H.265/HEVC video

For the integration in the context of DASH, the following applies for HEVC

- The encapsulation of HEVC video data in ISO BMFF is defined in ISO/IEC 14496-15 [10]. Clients shall support both sample entries ' using 'hvc1' and 'hev1', i.e.. inband Storage for VPS/SPS/PPS.
- Additional constraints within one Adaptation Set are provided in section 6.2.5.
- IDR pictures with `nal_unit_type` equal to `IDR_N_LP` and `IDR_W_RADL` are mapped to SAP types 1 and 2, respectively. BLA pictures with `nal_unit_type` equal to `BLA_N_LP` and `BLA_W_RADL` are mapped to SAP types 1 and 2, respectively. BLA pictures with `nal_unit_type` equal to `BLA_W_LP` are mapped to SAP type 2. CRA pictures with `nal_unit_type` equal to `CRA_NUT` are mapped to SAP type 3.
- The signaling of the different video codec profile and levels for the codecs parameters is according to ISO/IEC 14496-15 [10] Annex E. Note that any of the codecs present in Table 1 conforms to the profile level combination that is supported in DASH-HEVC.

  NOTE: For a detailed description on how to derive the signaling for the codec profile for H.264/AVC, please refer to DVB DASH, section 5.2.2.

Table 15 Codecs parameter according to ISO/IEC 14496-15 [10]

Profile	Level	Tier	Codec Parameter
HEVC Main	3.1	Main	hev1.1.2.L93.B0 hvc1.1.2.L93.B0
	4.1	Main	hev1.1.2.L123.B0 hvc1.12.L123.B0
HEVC Main-10	4.1	Main	hev1.2.4.L123.B0 hvc1.2.4.L123.B0

6.2.4. Video Metadata

The provisioning of video metadata in the MPD is discussed in section 3.2.4.

6.2.5. Adaptation Sets Constraints

6.2.5.1. General

Video Adaptation Sets shall contain Representations that are alternative encodings of the same source content. Video Adaptation Sets may contain Representations encoded at lower resolutions that are exactly divisible subsamples of the source image size. As a result, the cropped vertical and horizontal sample counts of all Representations can be scaled to a common display size without position shift or aspect ratio distortion that would be visible during adaptive switching. Subsample ratios must result in integer values for the resulting encoded sample counts (without rounding or truncation). The encoded sample count shall scale to the source video's exact active image aspect ratio when combined with the encoded sample aspect ratio value `aspect_ratio_idc` stored in the video Sequence Parameter Set NAL. Only the active video area shall be encoded so that devices can frame the height and width of the encoded video to the size and shape of their currently selected display area without extraneous padding in the decoded video, such as “letterbox bars” or “pillarbox bars”.

All decoding parameter sets referenced by NALs in a Representation using ‘avc1’ or ‘hvc1’ sample description shall be indexed to that track’s sample description table and decoder configuration record in the ‘avcC’ or ‘hvcC’ box contained in its Initialization Segment. All decoding parameter sets referenced by NALs in a Representation using ‘avc3’ or ‘hev1’ sample description shall be indexed to a Sequence Parameter NAL (SPS) and Picture Parameter NAL (PPS) stored prior to the first video sample in that Media Segment. For ‘avc3’ and ‘hev1’ sample description Representations, the SPS and PPS NALs stored in ‘avcC’ or ‘hvcC’ in the Initialization Segment shall only be used for decoder and display initialization, and shall equal the highest Tier, Profile, and Level of any SPS in the Representation. SPS and PPS stored in each Segment shall be used for decoding and display scaling.

For all Representations within an Adaptation Set with the following parameters shall apply.

- All the Initialization Segments for Representations within an Adaptation Set shall have the same sample description `codingname`. For example the inclusion of 'avc1' and 'avc3' based Representations within an Adaptation Set or the inclusion ‘avc1’ and ‘hev1’ based Representations within an Adaptation Set is not permitted.
- All Representations shall have equal timescale values in all `@timescale` attributes and ‘tkhd’ `timescale` fields in Initialization Segments.
- If ‘avc1’ or ‘hvc1’ sample description is signaled in the **AdaptationSet**`@codecs` attribute, an edit list may be used to synchronize all Representations to the presentation timeline, and the edit offset value shall be equal for all Representations.
- Representations in one Adaptation Set shall not differ in any of the following parameters: Color Primaries, Transfer Characteristics and Matrix Coefficients. If Adaptation Sets differ in any of the above parameters, these parameters should be signaled on Adaptation Set level. If signaled, a Supplemental or Essential Property descriptor shall be used, with the `@schemeIdURI` set to `urn:mpeg:mpegB:cicp:<Parameter>` as defined in ISO/IEC 23001-8 [45] and <Parameter> one of the following: `ColourPrimaries`, `TransferCharacteristics`, or `MatrixCoefficients`. The @value attribute shall be set as defined in ISO/IEC 23001-8 [45].

6.2.5.2. Bitstream Switching

For AVC and HEVC video data, if the `@bitstreamswitching` flag is set to true, then the following additional constraints shall apply:

- All Representations shall be encoded using ‘avc3’ sample description for AVC or ‘hev1’ for HEVC, and all IDR pictures shall be preceded by any SPS and PPS NAL decoding parameter referenced by a video NAL in that codec video sequence.

  Note: NAL parameter indexes in a Media Segment are scoped to that Segment. NALs and indexes in the Initialization Segment may be different, and are only used for decoder initialization, not Segment decoding.

- All Representations within a video Adaptation Set shall include an Initialization Segment containing an 'avcC' or 'hvcC' Box containing a Decoder Configuration Record containing SPS and PPS NALs that equal the highest Tier, Profile, Level, vertical and horizontal sample count of any Media Segment in the Representation. HEVC Decoder Configuration Records shall also include a VPS NAL.
- The **AdaptationSet**@codecs attribute shall be present and equal the maximum profile and level of any Representation contained in the Adaptation Set.
- The **Representation**@codecs attribute may be present and in that case shall equal the maximum profile and level of any Segment in the Representation.
- Edit lists shall not be used to synchronize video to audio and presentation timelines.
- Video Media Segments shall set the first presented sample's composition time equal to the first decoded sample's decode time, which equals the baseMediaDecodeTime in the Track Fragment Decode Time Box ('tfdt').

  Note: This requires the use of negative composition offsets in a v1 Track Run Box ('trun') for video samples, otherwise video sample reordering will result in a delay of video relative to audio.
- The **AdaptationSet**@presentationTimeOffset attribute shall be sufficient to align audio video, subtitle, and presentation timelines at presentation a Period's presentation start time. Any edit lists present in Initialization Segments shall be ignored. It is strongly recommended that the Presentation Time Offset at the start of each Period coincide with the first frame of a Segment to improve decoding continuity at the start of Periods.

NOTE: An Adaptation Set with the attribute **AdaptationSet**@bitstreamSwitching="true" fulfills the requirements of the DVB DASH specification [38].

See section 7.5 for additional Adaptation Set constraints related to content protection.

6.3. Audio

6.3.1. General

Content offered according to DASH-AVC/264 IOP is expected to contain an audio component in most cases. Therefore, clients consuming DASH-AVC/264-based content are expected to support stereo audio. Multichannel audio support and support for additional codecs is defined in extensions in section 9 of this document.

The codec for basic stereo audio support is MPEG-4 High Efficiency AAC v2 Profile, level 2 [12].

Notes

- HE-AACv2 is also standardized as Enhanced aacPlus in 3GPP TS 26.401 [14].
- HE-AACv2 Profile decoder [9] can also decode any content that conforms to
  - MPEG-4 AAC Profile [12]

- MPEG-4 HE-AAC Profile [12]

Therefore, Broadcasters and service providers encoding DASH-AVC/264 content are free to use any AAC version. It is expected that clients supporting the DASH-AVC/264 interoperability point will be able to play AAC-LC, HE-AAC and HE-AACv2 encoded content.

For all HE-AAC and HE-AACv2 bitstreams, explicit backwards compatible signaling should be used to indicate the use of the SBR and PS coding tools.

Note: To conform to the DVB DASH profile [38], explicit backwards compatible signaling shall be used to indicate the use of the SBR and PS coding tools.

For advanced audio technologies, please refer to section 9.

6.3.2. DASH-specific aspects for HE-AACv2 audio

In the context of DASH, the following applies for the High Efficiency AAC v2 Profile

- The content should be prepared according to the MPEG-DASH Implementation Guidelines [7] to make sure each (Sub)Segment starts with a SAP of type 1.
- The signaling of MPEG-4 High Efficiency AAC v2 for the codecs parameters is according to IETF RFC6381 [11] and is documented in Table 16. Table 16 also provides information on the ISO BMFF encapsulation.

Table 16 HE-AACv2 Codecs parameter according to RFC6381 [11]

Codec	Codec Parameter	ISO BMFF Encapsulation	SAP type
MPEG-4 AAC Profile [12]	`mp4a.40.2`	ISO/IEC 14496-14 [13]	1
MPEG-4 HE-AAC Profile [12]	`mp4a.40.5`	ISO/IEC 14496-14 [13]	1
MPEG-4 HE-AAC v2 Profile [12]	`mp4a.40.29`	ISO/IEC 14496-14 [13]	1

Note: Since both, HE-AAC and HE-AACv2 are based on AAC-LC, for the above-mentioned "Codec Parameter" the following is implied:

- `mp4a.40.5` = `mp4a.40.2` + `mp4a.40.5`
- `mp4a.40.29` = `mp4a.40.2` + `mp4a.40.5` + `mp4a.40.29`

6.3.3. Audio Metadata

6.3.3.1. General

Metadata for audio services is defined in ISO/IEC 23009-1.

6.3.3.2. ISO/IEC 23009-1 audio data

With respect to the audio metadata, the following elements and attributes from ISO/IEC 23009-1 are relevant:

- the `@audioSamplingRate` attribute for signaling the sampling rate of the audio media component type in section 5.3.7 of ISO/IEC 23009-1
- the **`AudioChannelConfiguration`** element for signaling audio channel configuration of the audio media component type.in section 5.3.7 of ISO/IEC 23009-1.

6.4. Auxiliary Components

6.4.1. Introduction

Beyond regular audio and video support, TV programs typically also require support for auxiliary components such as subtitles and closed captioning, often due to regulatory requirements. DASH-AVC/264 provides tools to addresses these requirements.

6.4.2. Subtitles and Closed Captioning

Technologies for subtitles are as follows:

- CEA-708 Digital Television (DTV) Closed Captioning [15]
- W3C TTML [17] and the SMPTE profile on SMPTE Timed Text [18]. Graphics-based subtitles and closed captioning are also supported by SMPTE Timed Text [18].
- 3GPP Timed Text [16]
- Web VTT [19]
- W3C subtitles Internet Media and Subtitles [19].

For simple cases, CEA-608/708 based signaling as defined in section 6.4.3 may be used.

For any other use cases, W3C TTML [17] and the SMPTE profile on SMPTE Timed Text [18] should be used as defined in section 6.4.4.

6.4.3. CEA-608/708 in SEI messages

6.4.3.1. Background

In order to provide the signaling of the presence of SEI-based data streams and closed captioning services on MPD level, descriptors on DASH level are defined. This section provides some background.

> Note: This method is compatible with draft SCTE specification and therefore SCTE URNs are used for the descriptor `@schemeIdURI`. In an updated version of this document more details on the exact relation to the SCTE specification will be provided.

The presence of captions and their carriage within the SEI message of a video track is defined in ANSI/SCTE 128-1 2013 [39], section 8.1 Encoding and transport of caption, active format description (AFD) and bar data.

Based on this it is enabled that a video track carries SEI message that carry CEA-608/708 CC. The SEI message `payload_type=4` is used to indicates that Rec. ITU-T T.35 based SEI messages are in use.

In summary the following is included in ANSI/SCTE 128-1 2013 to signal CEA-608/708 CC:

- SEI `payloadType` is set to 4
- `itu_t_t35_country_code` – A fixed 8-bit field, the value of which shall be `0xB5`.
- `itu_t_35_provider_code` – A fixed 16-bit field registered by the ATSC. The value shall be `0x0031`.
- `user_identifier` – This is a 32 bit code that indicates the contents of the `user_structure()` and is `0x47413934` ("GA94").
- `user_structure()` – This is a variable length data structure `ATSC1_data()` defined in section 8.2 of ANSI/SCTE 128 2013-a.
- `user_data_type_code` is set to `0x03` for indicating captioning data in the `user_data_type_structure()`
- `user_data_type_structure()` is defined in section 8.2.2 of ANSI/SCTE 128 2013 for Closed Captioning and defines the details on how to encapsulate the captioning data.

The semantics of relevant Caption Service Metadata is provided in CEA-708 [15], section 4.5:

- the total number of caption services (1-16) present over some transport-specific period.
- For each service:
  - The type of the service, i.e. being 608 or 708. According to CEA-708 [15], section 4.5, there shall be at most one CEA-608 data stream signaled. The CEA-608 datastream itself signals the individual CEA-608-E caption channels.
  - When the type of the service is 708, then the following 708-related metadata should be conveyed:
    - SERVICE NUMBER: the service number as found on the 708 caption service block header (1-31). This field provides the linkage of the remaining metadata to a specific 708 caption service
    - LANGUAGE: the dominant language of the caption service, recommended to be encoded from ISO 639.2/B [41].
    - DISPLAY ASPECT RATIO {4:3, 16:9}: The display aspect ratio assumed by the caption authoring in formatting the caption windows and contents.
    - EASY READER: this metadata item, when present, indicates that the service contains text tailored to the needs of beginning readers.

6.4.3.2. MPD-based Signaling of SEI-based CEA-608/708 Closed Caption services

This subsection provides methods MPD-based Signaling of SEI-based CEA-608/708 Closed Caption services, i.e.

- The presence of one or several SEI-based closed caption services in a Representation.
- The signaling of the relevant Caption Service Metadata as defined in CEA-708 [15], section 4.5.

The descriptor mechanism in DASH is used for this purpose.

Signaling is provided by including **`Accessibility`** descriptors, one each for CEA 608 and CEA 708 and is described in sections 6.4.3.3 and 6.4.3.4, respectively. The **`Accessibility`** descriptor is included for the **`AdaptationSet`** and all included Representations shall provide equivalent captions.

The `@value` attribute of each descriptor can be either list of languages or a complete map of services (or CC channels, in CEA-608 terminology). Listing languages without service or channel information is strongly discouraged if more than one caption service is present.

6.4.3.3. Signaling CEA-608 caption service metadata

The **`Accessibility`** descriptor shall be provided with `@schemeIdURI` set to `urn:scte:dash:cc:cea-608:2015`, and an optional `@value` attribute to describe the captions. If the `@value` attribute is not present, the Representation contains a CEA-608 based closed captioning service.

If present, the `@value` attribute shall contain a description of caption service(s) provided in the stream as a list of channel-language pairs. Alternatively, a simple list of language codes may be provided, but this is strongly discouraged as it will not provide sufficient information to map the language with the appropriate caption channel.

The `@value` syntax shall be as described in the ABNF below.

```
@value          = (channel *3 [";" channel]) / (language *3[";" language])
channel         = channel-number "=" language
channel-number  = CC1 | CC2 | CC3 | CC4
language        = 3ALPHA ; language code per ISO 639.2/B [41]
```

6.4.3.4. Signaling CEA-708 caption service metadata

The **`Accessibility`** descriptor shall be provided with the `@schemeIdURI` set to `urn:scte:dash:cc:cea-708:2015`, and an optional `@value` attribute to describe the captions. If the `@value` attribute is not present, the Representation contains a CEA-708 based closed captioning service.

If present, the `@value` shall contain the Caption Service Metadata as provided in CEA-708 [15], section 4.5 as a semicolon-separated string of service descriptions. Each service description is a list of colon-separated name-value pairs. Alternatively, a simple list of language codes may be provided, but this is strongly discouraged as it will not provide sufficient information to map the language with the appropriate caption service. The `@value` syntax shall be as described in the ABNF below.

```
@value              = service *15 [";" service] / (language *15[";" language])
service             = service-number "=" param
```

```
service-number    = (%d1 - %d63) ; decimal numbers 1 through 63
param             = "lang" ":" language[","easy-reader]["","aspect-ratio]
language          = 3ALPHA; language code per ISO 639.2/B
easy-reader       = "er" ":" BIT ; default value 0
wide-aspect-ratio = "war" ":" BIT ;  default value 1 (16:9), 0 for 4:3
```

NOTE: `ALPHA` and `BIT` are as defined by IETF RFC 5234 [43], Appendix B.1.

The following parameters are defined for signaling CEA-708 metadata:

Name	Value	Comments
`lang`	LANG	LANGUAGE code (see above)
`war`	Aspect ratio of the service.	The value may be either "4:3" (set to 0) or "16:9" (set to 1).
`er`	EasyReader presence ('1' if present, '0' if not).	If not present, 0 is assumed

Each of the `Service` parameters (except for language) may be present or not present. Default values can be assumed where specified.

6.4.3.5. Examples

Simple signaling of presence of CEA-608 based closed caption service (Note: Not signaling languages is a discouraged practice)

```
<Accessibility
    schemeIdUri="urn:scte:dash:cc:cea-608:2015"/>
```

Signaling of presence of CEA-608 closed caption service languages in English and German

```
<Accessibility
    schemeIdUri="urn:scte:dash:cc:cea-608:2015"
    value="eng;deu"/>
```

Signaling of presence of CEA-608 closed caption service in English and German, with channel assignments

```
<Accessibility
    schemeIdUri="urn:scte:dash:cc:cea-608:2015"
    value="CC1=eng;CC3=deu"/>
```

Signaling of presence of CEA-708 closed caption service in English and German

```
<Accessibility
    schemeIdUri="urn:scte:dash:cc:cea-708:2015"
    value="1=lang:eng;2=lang:deu"/>
```

Signaling of presence of CEA-708 closed caption service in English and easy reader English

```
<Accessibility
    schemeIdUri="urn:scte:dash:cc:cea-708:2015"
    value="1=lang:eng;2=lang:eng,war:1,er:1"/>
```

6.4.4. SMPTE Timed Text

W3C TTML [17] and the SMPTE profile on SMPTE Timed Text [18] provide a rich feature set for subtitles. Beyond basic subtitles and closed captioning, for example, graphics-based subtitles and closed captioning are also supported by SMPTE Timed Text [18]. Conversion of CEA-608 and CEA-708 into SMPTE TT may be done according to SPMTE 2052-10 [23] and SMPTE-2052-11 [24], respectively. Note that by the choice of SMPTE TT as the supported format at the client, other formats such as EBU TT [21] are also supported as long as only the subset that is also supported by SMPTE TT is used in the content authoring.

In the context of DASH, the following applies for text/subtitling:

- All graphics type samples are SAP type 1.
- The signalling of the different text/subtitling codecs for the codecs parameters is according to RFC6381 [11] and is documented in Table 17. Table 17 also provides information on ISO BMFF encapsulation.
- For live services, encapsulation in ISO BMFF is definitely necessary. However, for On-Demand cases, the full file of subtitles may be provided as XML data only.

Table 17 Subtitle Codecs parameter according to RFC6381 [11]

Codec	MIME type	Codec Parameter @codecs	ISO BMFF Encapsulation
SMPTE Timed Text [18] without encapsulation	`application/ttml+xml` (1,3)	`not present`	n/a
SMPTE Timed Text [18] with ISO BMFF encapsulation	`application/mp4`	`stpp` (2,3)	ISO/IEC 14496-12 [8] ISO/IEC 14496-30 [25]

Notes:

(1) the MIME type is generic for TTML based timed text and not specific to SMPTE-TT (for example, DVB DASH is using the same MIME type to signal EBU-TT). For details on the used format, parsing of the XML document is necessary. Nevertheless, it is expected that a generic TTML-based parser can decode and render any type of TTML-based subtitles

(2) the sample entry **`'stpp'`** is generic for TTML based timed text and not specific to SMPTE-TT (for example, DVB DASH is using 'stpp' to signal EBU-TT). For details on the used format, parsing

of the subtitle sample entry **`schema_location`** parameter ís necessary. Nevertheless, it is expected that a generic TTML-based parser can decode and render any type of TTML-based subtitles.

(3) DVB DASH only supports ISO BMFF encapsulated TT, but not XML-based.

6.4.5. Annotation of Subtitles

Subtitles should be annotated properly using descriptors available in ISO/IEC 23009-1, Specifically Role, Accessibility, Essential Property and Supplemental Property descriptors and the DASH role scheme may be used. Guidelines for annotation are for example provided in DVB DASH, section 7.1.2 or SCTE-DVS 1202, section 7.2.

7. Content Protection Related Aspects

7.1. Introduction

DASH-AVC/264 does not intend to specify a full end-to-end DRM system. However DASH-AVC/264 provides a framework for multiple DRMs to protect DASH content by adding instructions or *Protection System Specific*, proprietary information in predetermined locations in MPDs, or DASH content that is encrypted with Common Encryption as defined in ISO/IEC 23001-7 [26].

The Common Encryption (‘`cenc`’) protection scheme specifies encryption parameters that can be applied by a scrambling system and key mapping methods using a common key identifier (KID) to be used by different DRM systems such that the same encrypted version of a file can be combined with different DRM systems that can store proprietary information for licensing and key retrieval in the Protection System Specific Header Box (‘`pssh`’), or in **`ContentProtection`** Descriptors in an MPD. The DRM scheme for each `pssh` is identified by a DRM specific `SystemID`.

The recommendations in this document reduce the encryption parameters and use of the encryption metadata to specific use cases for VOD and live content with key rotation.

The base technologies are introduced first followed by informative chapter on standardized elements. Additional Content Protection Constraints are then listed that are specific to conformance to DASH-264/AVC IOP.

7.2. Base Technologies Summary

The normative standard that defines common encryption in combination with ISO BMFF is ISO/IEC 23001-7:2014 2nd Ed., CENC [26]. It includes:

- Common Encryption of NAL structure video and other media data with AES-128 CTR mode
- Support for decryption of a single Representation by multiple DRM systems
- Key rotation (changing media keys over time)
- XML syntax for expressing a default KID attribute and pssh element in MPDs

The main DRM components are:

1. The **ContentProtection** descriptors in the MPD (see [5] 5.3.7.2-Table 9, 5.8.5.2 and [5][1] 5.8.4.1) that contains the URI for signaling of the use of Common Encryption or the specific DRM being used.
2. 'tenc' parameters that specify encryption parameters and default_KID (see [26] 8.2.1). The 'tenc' information is in the Initialization Segment. Any KIDs in Movie Fragment sample group description boxes override the 'tenc' parameter of the default_KID, as well as the 'not encrypted' parameter. Keys referenced by KID in sample group descriptions must be available when samples are available for decryption, and may be stored in a protection system specific header box ('pssh') in each movie fragment box ('moof'). The default_KID information may also appear in the MPD (see [26] 11.1).
3. 'senc' parameters that may store initialization vectors and subsample encryption ranges. The 'senc' box is stored in each track fragment box ('traf') of an encrypted track (see [21] 7.1), and the stored parameters accessed using the sample auxiliary information offset box ('saio') and the sample auxiliary information size box ('saiz') (see [4] 8.7.8 and 8.7.9).
4. 'pssh' license acquisition data or keys for each DRM in a format that is "Protection System Specific". 'pssh' refers to the Protection System Specific Header box described in [26], 8.1.2. 'pssh' boxes may be stored in Initialization or Media Segments (see [26] 8.1 and 8.2). It may also be present in a cenc:pssh element in the MPD (see [5] 5.8.4.1, [26] 11.2.1). cenc:pssh information in the MPD increases the MPD size but may allow faster parsing, earlier access and addition of DRMs without content modification.
5. Key rotation is mainly used to allow changes in entitlement for continuous live content. It is used as defined in [26] with the following requirements:
    - In the initialization segment, the movie box ('moov') contains a Track Encryption Box ('tenc') and may contain a 'pssh' box for each DRM e.g. to store root license acquisition information for authentication and authorization.
    - Sample To Group Box ('sbgp') and Sample Group Description Box ('sgpd') of type 'seig' are used to indicate the KID applied to each sample, and changes to KIDs over time (i.e. "key rotation"). (see [4] 8.9.4) KIDs referenced by sample groups must have the keys corresponding to those KIDs available when the samples in a Segment are available for decryption. Keys referenced by sample groups in a Segment may be stored in that Segment in Protection System Specific Header Boxes ('pssh') stored in the Movie Fragment Box ('moof'). A version 1 'pssh' box may be used to list the KID values stored to enable removal of duplicate boxes if a file is defragmented.
    - Keys stored in Media Segment 'pssh' boxes must be stored in the same DRM format for all users so that the same Media Segments can be shared by all users. User-specific information must be delivered "out of band", as in a "root" license associated with the default_KID, which can be individualized for each DRM

client, and control access to the shared `pssh` information stored in Media Segments, e.g. by encrypting the keys stored in Segment `pssh` boxes with a "root key" provided by the user-specific DRM root license. Common Encryption specifies `pssh` to enable key storage in movie fragments/Segments; but it does not preclude other methods of key delivery that satisfy KID indexing and availability requirements.

7.3. ISO BMFF Support for Common Encryption and DRM

The ISO Media Format carries content protection information in different locations. Their hierarchy is explained in the informational chapter below, followed by a reference on where these elements are standardized.

7.3.1. Box Hierarchy

The following shows the box hierarchy and composition for relevant boxes, when using common encryption:

- `moov/pssh` (zero or one per system ID)
- `moov/trak/mdia/minf/stbl/stsd/sinf/schm` (one, if encrypted)
- `moov/trak/mdia/minf/stbl/stsd/sinf/schi/tenc` (one, if encrypted)
- `moof/traf/saiz` (one, if encrypted)
- `moof/traf/saio` (one, if encrypted)
- `moof/traf/senc` (one, if encrypted)

for key rotation

- `moof/traf/sbgp` (one per sample group)
- `moof/traf/sgpd 'seig'(sample group entry)` (one per sample group)
- `moof/pssh` (zero or one per system ID)

Graphical overview of above structure for VOD content is shown in the figure below

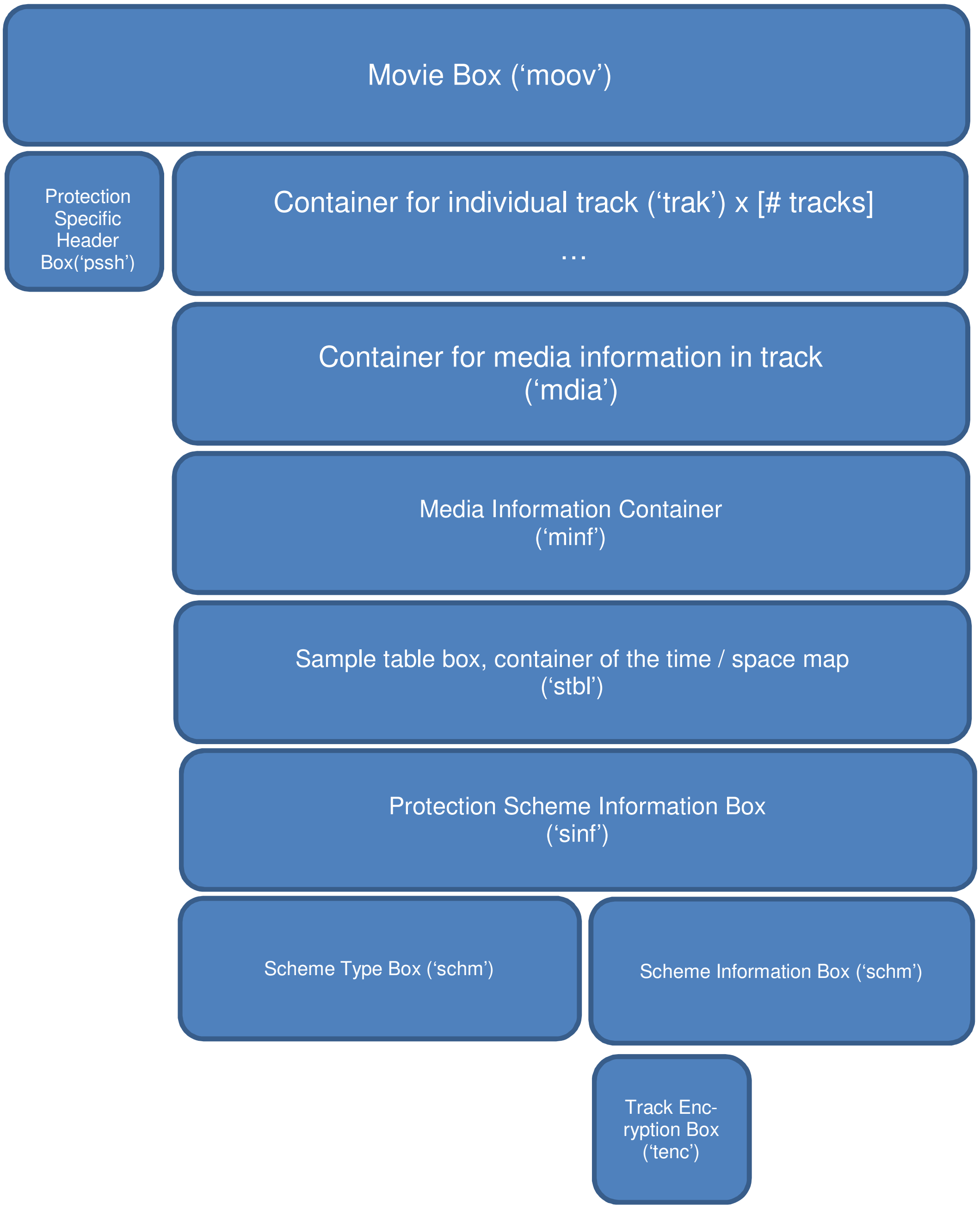


Figure 17: Visualization of box structure for single key content

Graphical overview of box structure for live content is shown in the figure below

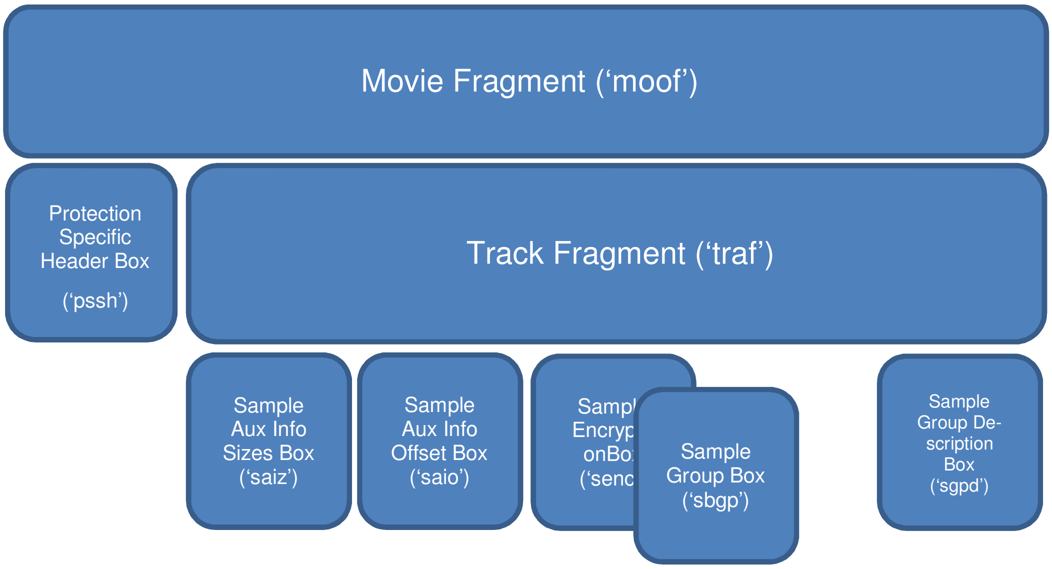


Figure 18: Visualization of box structure with key rotation

7.3.2. ISO BMFF Structure Overview

below provides pointers to relevant information in the specifications to understand the standard DRM components and if the main description is located in the ISO base media file format ([8]), or the Common Encryption specification ([26]).

Table 18 Boxes relevant for DRM systems

Box	**Full Name / *Usage***	**Reference**
`moof`	movie fragment header *One 'moof' box for each fragment, i.e. Media Segment/Subsegment.*	[8], 8.32 + [5]
`moov`	movie header, container for metadata *One 'moov' box per file.*	[8] , 8.1
`pssh`	Protection System Specific Header Box *Contains DRM specific data. pssh box version 1 (specified in Common Encryption 2nd edition) contains a list of KIDs to allow removing duplicate 'pssh' boxes when defragmenting a file by comparing their KIDs*	[26], 8.1.1
`saio`	Sample Auxiliary Information Offsets Box *Contains the offset to the IVs & subsample encryption byte ranges.*	[8], 8.7.9
`saiz`	Sample Auxiliary Information Sizes Box *Contains the size of the IVs & subsample encryption byte ranges.*	[8], 8.7.8
`senc`	Sample Encryption Box *Contains Initialization Vectors; and subsample ranges for a Media Segment*	[21] 7.1
`schi`	Scheme Information Box *Container boxes used by that protection scheme type.*	[8], 8.12.6 + [26], 4

`schm`	Scheme Type Box *Contains the encryption scheme, identified by a 4 character code, e.g. 'cenc'*	[8] , 8.12.5 + [26], 4
`seig`	Cenc Sample Encryption Information Video Group Entry *A sample description containing KIDs describing sample groups in this segment, for key rotation.*	[26], 6
`sbgp`	Sample to Group Box *lists a group of samples*	[8] , + [26], 5
`sgpd`	Sample Group Description Box *Describes properties of a sample group*	[8] , 8.9.3 + [26], 5
`sinf`	Protection Scheme Information Box *Signals that the stream is encrypted*	[8] , 8.12.1 + [26], 4
`stsd`	Sample description table (codec type, initialization parameters, stream layout, etc.)	[8], 8.16
`tenc`	Track Encryption Box *Contains default encryption parameters for the entire track, e.g. default_KID*	[26], 8.2.1

7.4. MPD support for Encryption and DRM Signaling

The MPD contains signaling of the content encryption and key management methods used to help the receiving client determine if it can possibly play back the content. The MPD elements to be used are the **`ContentProtection`** Descriptor elements. At least one **`ContentProtection`** Descriptor element SHALL be present in each **`AdaptationSet`** element describing encrypted content.

7.4.1. Use of the Content Protection Descriptor

7.4.1.1. `ContentProtection` Descriptor for *mp4protection* Scheme

A **`ContentProtection`** descriptor with the `@schemeIdUri` value of `"urn:mpeg:dash:mp4protection:2011"` signals that the content is encrypted with the scheme indicated in the `@value` attribute. The file structure of content protection schemes is specified in [8], 5.8.5.2, and the `@value` = '`cenc`' for the Common Encryption scheme, as specified in [21]. Although the **`ContentProtection`** Descriptor for UUID Scheme described below is usually used for license acquisition, the **`ContentProtection`** Descriptor with `@schemeIdUri="urn:mpeg:dash:mp4protection:2011"` and with `@cenc:default_KID` may be sufficient to acquire a license or identify a previously acquired license that can be used to decrypt the Adaptation Set. It may also be sufficient to identify encrypted content in the MPD when combined with license acquisition information stored in 'pssh' boxes in Initialization Segments.

A **`ContentProtection`** Descriptor for the mp4 Protection Scheme is used to identify the default `KID`, as specified by the '`tenc`' box, using the `@cenc:default_KID` attribute defined in [26], section 11.1. The value of the attribute is the `KID` expressed in UUID string notation.

```
<ContentProtection schemeIdUri="urn:mpeg:dash:mp4protection:2011"
        value="cenc" cenc:default_KID="34e5db32-8625-47cd-ba06-68fca0655a72"/>
```

When the `default_KID` is present on each Adaptation Set, it allows a player to determine if a new license needs to be acquired for each Adaptation Set by comparing their `default_KIDs` with each other, and with the `default_KIDs` of stored licenses. A player can simply compare these KID strings and determine what unique licenses are necessary without interpreting license information specific to each DRM system.

7.4.1.2. ContentProtection Descriptor for UUID Scheme

A UUID **`ContentProtection`** descriptor in the MPD may indicate the availability of a particular DRM scheme for license acquisition. An example is provided below:

```
<ContentProtection
    schemeIdUri="urn:uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    value="DRMNAME version"/>
```

The schemeIdUri uses a UUID URN with the UUID string equal to the registered SystemID for a particular DRM system. A list of known DRM SystemIDs can be found in the DASH identifier repository available here: `http://www.dashif.org/identifiers/content-protection`.

This is specified in [8], 5.8.5.2 and is referred to as "**`ContentProtection`** Descriptor for UUID Scheme" in the following.

7.4.1.3. Protection System Specific Header Box `cenc:pssh` element in MPD

A '`pssh`' box is defined by each DRM system for use with their registered `SystemID`, and the same box can be stored in the MPD within a **`ContentProtection`** Descriptor for UUID scheme using an extension element in the "`cenc:`" namespace. Examples are provided in [7] and in [26] sec. 11.2.

Carrying `cenc:default_KID` attribute and a `cenc:pssh` element in the MPD is useful to allow key identification, license evaluation, and license retrieval before live availability of initialization segments. This allows clients to spread license requests and avoid simultaneous requests from all viewers at the instant that an Initialization Segments containing license acquisition information in 'pssh' becomes available. With `cenc:default_KID` indicated in the mp4protection **`ContentProtection`** Descriptor on each Adaptation Set, clients can determine if that key and this presentation is not available to the viewer (e.g. without purchase or subscription), if the key is already downloaded, or which licenses the client SHOULD download before the `@availabilityStartTime` of the presentation based on the `default_KID` of each AdaptationSet element selected.

7.5. Additional Content Protection Constraints

The following describes additional constraints for presentations to be conformant with DASH-264/AVC, for both MPD and ISO Media files:

7.5.1. ISO BMFF Content Protection Constraints

- There SHALL be identical values of `default_KID` in the Track Encryption Box (‘`tenc`’) of all Representation referenced by one Adaptation Set. Different Adaptation Sets may have equal or different values of `default_KID`.
- In the case where ‘`pssh`’ boxes are present in Initialization Segments, each Initialization Segment within one Adaptation Set SHALL contain an equivalent pssh box for each `SystemID`, i.e. license acquisition from any Representation is sufficient to allow switching between Representations within the Adaptation Set without acquiring a new license.
- For “key rotation”, each Movie Fragment SHOULD contain one ‘`pssh`’ in each ‘`moof`’ box per `SystemID` that contains sufficient information for the DRM system with matching `SystemID` to obtain protected keys for this movie fragment, when combined with:
    - information from ‘`pssh`’ in ‘`moov`’ or `cenc:pssh` in MPD.
      This may be used to download a Root license bound to this device, necessary to read the key rotation Segment licenses common to all Media Segments in the case where the DRM is using a root and leaf license hierarchy.
      Note that the presence and use of root and leaf licenses is optional and DRM specific.
    - `KID` associated with each sample from ‘`seig`’ sample group description box
    - Sample to group boxes that list all the samples that use a particular `KID`

  Note: A ‘pssh’ for each `SystemID` in each ‘`moof`’ will likely result in duplicate ‘`pssh`’ boxes containing some of the same `KID`s, but it facilitates random access into live presentations, and trick play of linear content with a PVR buffer or when made available as VOD assets.

7.5.2. MPD Content Protections Constraints

- For an encrypted Adaptation Set, ContentProtection Descriptors SHALL always be present in the `AdaptationSet` element, and apply to all contained Representations.
- A ContentProtection Descriptor for the mp4 Protection Scheme with the `@schemeIdUri` value of `"urn:mpeg:dash:mp4protection:2011"` and `@value='cenc'` SHALL be present in the AdaptationSet element if the contained Representations are encrypted.

  Note that this allows clients to recognize the Adaptation Set is encrypted with common encryption scheme without the need to understand any system specific UUID descriptors.

  The ContentProtection Descriptor for the mp4protection scheme SHOULD contain the optional attribute `@cenc:default_KID`. The ‘`tenc`’ box that specifies the encoded track encryption parameters SHALL be considered the source of truth for the default key ID value since it contains the `default_KID` field, and is present in the movie box, as specified in [26], section 8.2.1. The MPD cenc:default_KID attribute SHALL match the 'tenc' `default_KID`.

Note that this allows clients to identify the default KID from the MPD using a standard location and format, and makes it accessible to general purpose clients that don't understand the system specific information formats of all DRM schemes that might be signaled.

- The `cenc:pssh` element SHOULD be present in the ContentProtection Descriptor for each UUID Scheme. The base64 encoded contents of the element SHALL be equivalent to a `'pssh'` box including its header. The information in the `'pssh'` box SHOULD be sufficient to allow for license acquisition.

Note: A player such as DASH.js hosted by a browser may pass the contents of this element through the Encrypted Media Extension (EME) API to the DRM system Content Decryption Module (CDM) with a SystemID equal to the Descriptor's UUID. This allows clients to acquire a license using only information in the MPD, prior to downloading Segments.

Below is an example of the recommended format for a hypothetical acme DRM service:

```
<ContentProtection schemeIdUri="urn:uuid:d0ee2730-09b5-459f-8452-200e52b37567"
      value="Acme DRM 2.0">

      <!-- base64 encoded 'pssh' box with SystemID matching the containing ContentProtection Descriptor -->
      <cenc:pssh>
        YmFzZTY0IGVuY29kZWQgY29udGVudHMgb2YgkXB
        zc2iSIGJveCB3aXRoIHRoaXMgU3lzdGVtSUQ=
      </cenc:pssh>

</ContentProtection>
```

- The `@value` attribute of the ContentProtection Descriptor for UUID Scheme SHOULD contain the DRM system and version in a human readable form.

7.5.3. Other Content Protections Constraints

In the case where the '`pssh`' box element is present in the MPD and in the Initialization Segment, the '`pssh`' box element in the MPD SHALL take precedence, because the parameters in the MPD will be processed first, are easier to update, and can be assumed to be up to date at the time the MPD is fetched.

Recommended scheduling of License and key delivery:

- Request licenses on initial processing of an MPD if ContentProtection Descriptors or Initialization Segments are available with license acquisition information. This is intended to avoid a large number of synchronized license requests at MPD@availabilityStartTime.
- Prefetch licenses for a new Period in advance of its presentation time to allow license download and processing time, and prevent interruption of continuous decryption and playback. Advanced requests will also help prevent a large number of synchronized license requests during a live presentation at Period@start time.

- Store keys referenced by Sample Groups in each Segment that references them (in 'moof'/'pssh'), or deliver referenced keys before those samples may be accessed to allow continuous sample decryption and decoding during random access to all available Segments.
- Each Segment (movie fragment) SHOULD be limited to a single KID and sample group to simplify the representation and processing of key rotation.

7.5.4. Encryption of Different Representations

Representations contained in one Adaptation Set SHALL be protected by the same license for each protection system ("DRM"), and SHALL have the same value of '`default_KID`' in their '`tenc`' boxes in their Initialization Segments. This is to enable seamless switching within Adaptation Sets, which is generally not possible if a new DRM license needs to be authorized, client bound, generated, downloaded, and processed for each new Representation.

In the case of key rotation, if root licenses are used, the same requirement applies to the root licenses (one license per Adaptation Set for each DRM), and also means all Representations SHALL have the same value of '`default_KID`' in their '`tenc`' boxes in their Initialization Segments. The use of root and leaf licenses is optional and DRM specific, but leaf licenses are typically delivered in band to allow real time license acquisition, and do not require repeating client authentication, authorization, and rebuilding the security context with each key change in order to enable continuous playback without interruption cause be key acquisition or license processing.

In cases where SD and HD and UHD Representations are contained in one presentation, different license rights may be required for each quality level and may be sold separately. If different licenses are required for different quality levels, then it is necessary to create separate Adaptation Sets for each quality level, each with a different license and value of '`default_KID`'.

Representations that are equivalent resolution and bitrate but encrypted with different keys may be included in different Adaptation Sets. Seamless switching between UHD, HD and SD Representations is difficult because these quality levels typically use different decryption licenses and keys, use different DRM output rules (prohibit analog interfaces, require resolution down-scaling, require HDCP encryption on output, etc.), and use different decoding parameters for e.g. subsampling, codec, profile, bit depth, aspect ratios and color spaces.

If any Representation is encrypted in an Adaptation Set, then all must be encrypted using the same `default_KID` in the Track Encryption Box ('`tenc`') to avoid realtime changes to the DRM licenses and security context. KID values may change over time ("key rotation") as specified in Common Encryption and a particular DRM system.

For all Representations within an Adaptation Set with `@bitstreamSwitching`="`false`" (default), the following parameters shall apply.

- '`tenc`' `default_KID` shall be equal for all Representations

7.5.5. Encryption of Multiple Periods

Periods SHALL only change default_KID and corresponding license at the start of a new Period corresponding to a different file. A different file is indicated by a different default_KID signaled in the 'tenc' box in the Initialization Segment.

A file associated with a single license may be continued over multiple Periods by being referenced by multiple Representations over multiple Periods (for instance, a program interspersed with ad Periods). A client can recognize the same cenc:default_KID value and avoid having to download the same license again; but the DRM system may require a complete erase and rebuild of the security context, including all key material, samples in process, etc., between Periods with different licenses or no license (between protected and clear Periods).

7.5.6. DRM System Identification

The DRM system is signaled in the MPD and `'pssh'` boxes with a `SystemID`. A list of known DRMs can be found in the DASH identifier repository available here: `http://www.dashif.org/identifiers/content-protection`.

7.5.7. Protection of Media Presentations that Include SD, HD and UHD Adaptation Sets

Per DASH IF interop points, Representations with separate keys, licenses, and license policy are contained in different Adaptation Sets.

Adaptive bitrate switching can function automatically within an Adaptation Set without changing keys, licenses, robustness and output rules, etc.

A player may download licenses for multiple Adaptation Sets in a Group, and seamlessly switch between them if it is able. Seamless switching between Adaptation Sets is allowed, but not required. DASH may need to signal which Adaptation Sets are intended for seamless switching, i.e. have identical source content, same picture aspect ratio, same exact rescaled pixel registration, same sample description (e.g. `'avc3'`), same initialization behavior (`@bitstreamSwitching = true/false`), same Timescale and `@timescale`, and are mutually time-aligned.

The DASH-IF interop points are intended to make bitrate switching within an Adaptation Set simple and automatic, whether Representations are encrypted or not. Placement of Representations in different Adaptation Sets informs players that those Representations need to be initialized with different parameters, such as a different key and license. The full initialization process is repeated per Period. Adaptation Sets with `@bitstreamSwitching` = "`true`" only need to be initialized once per Period. Adaptation Sets with `@bitstreamSwitching = "false`" need to be partially re-initialized on each Representation switch (to change the SPS parameter sets referenced from NALs to those stored in in the containing track's 'avcC'), but most initialized parameters such as timescale, codec Profile/Level, display buffer size, colorspace, etc.; and licenses and the DRM system … do not need to be changed.

Fetching and resetting keys and licenses during adaptive switching requires processing Initialization Segments with different `'tenc'` default_KID and possibly `'pssh'` boxes. That may not be seamless, especially in browser playback where the decoders are only aware of player switching when an Initialization Segment flows through the MSE buffer and a `needKey()` event is raised via EME.

Note that switching between Adaptation Sets with different Media Profiles could be restricted by key and license policy, e.g. the user only purchased SD rights, the player only has analog output and HD content requires a protected digital output, UHD content requires hardware protected DRM, etc.

Implementations that seamlessly switch between Representations with different keys and policies generally require a standardized presentation ID or content ID system that associates multiple keys and licenses to that ID and presentation, then downloads only the keys/licenses authorized for that user and device (e.g. SD or HD+SD). The player must then install those licenses and use player logic to select only Representations in an Adaptation Set for which a license is installed and output controls, display configuration, etc. allow playback (e.g. only Representations keyed for an installed SD license). Players and license servers without this pre-configuration protocol and adaptive switching logic will encounter key/license requests in the process of adaptive switching, and may find output blocked by different license policies, user rights, etc.

7.6. Workflow Overview

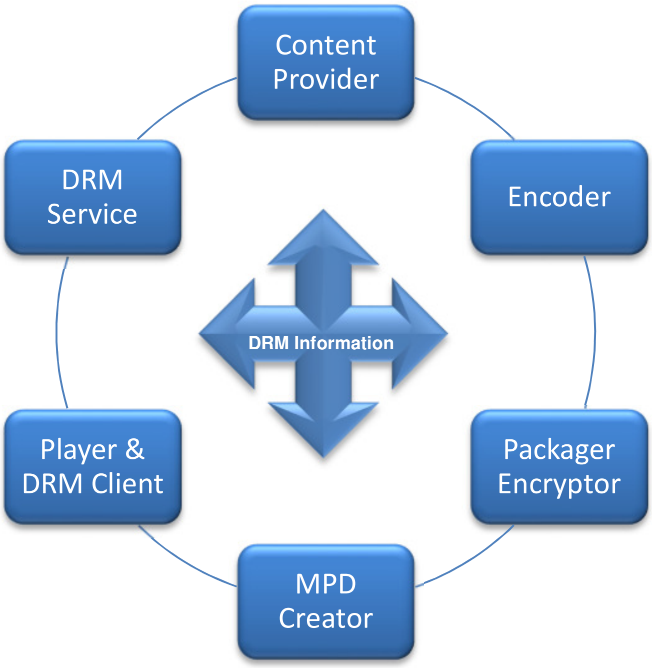


Figure 19 Logical Roles that Exchange DRM Information and Media

Figure 19 above shows logical entities that may send or receive DRM information such as media keys, asset identifiers, licenses, and license acquisition information. A physical entity may combine multiple logical roles, and the point of origin for information, such as media keys and asset identifiers, can differ; so various information flows are possible. This is an informative example of how the roles are distributed to facilitate the description of workflow and use cases. Alternative roles and functions can be applied to create conformant content.

Description of Logical Roles:

Content Provider – A publisher who provides the rights and rules for delivering protected media, also possibly source media (mezzanine format, for transcoding), asset identifiers, key identifiers (KID), key values, encoding instructions, and content description metadata.

Encoder – A service provider who encodes Adaptation Sets in a specified media format, number of streams, range of bitrates and resolutions, seamless switching constraints, etc., possibly determined by the publisher. An asset identifier needs to be assigned to each encoded track in order to associate a key identifier, a Representation element in an MPD, a possible ‘pssh’ box in the file header, and a DRM license separately downloaded.

Packager / Encryptor – A service provider who encrypts and packages media files, inserting default_KID in the file header ‘`tenc`’ box, initialization vectors and subsample byte ranges in track fragments indexed by ‘`saio`’ and ‘`saiz`’ boxes, and possibly packages ‘`pssh`’ boxes containing license acquisition information (from the DRM Provider) in the file header. Tracks that are partially encrypted or encrypted with multiple keys require sample to group boxes and sample group description boxes in each track fragment to associate different KIDs to groups of samples. The Packager could originate values for KIDs, media keys, encryption layout, etc., then send that information to other entities that need it, including the DRM Provider and Streamer, and probably the Content Provider. However, the Packager could receive that information from a different point of origin, such as the Content Provider or DRM Provider.

MPD Creator – The MPD Creator is assumed to create one or more types of DASH MPD, and provide indexing of Segments and/or '`sidx`' indexes for download so that players can byte range index Subsegments. The MPD must include descriptors for Common Encryption and DRM key management systems, and SHOULD include identification of the default_KID for each AdaptationSet element, and sufficient information in UUID ContentProtection Descriptor elements to acquire a DRM license. The default_KID is available from the Packager and any other role that created it, and the DRM specific information is available from the DRM Provider.

Player / DRM Client – Gets information from different sources: MPD, Media files and DRM License.

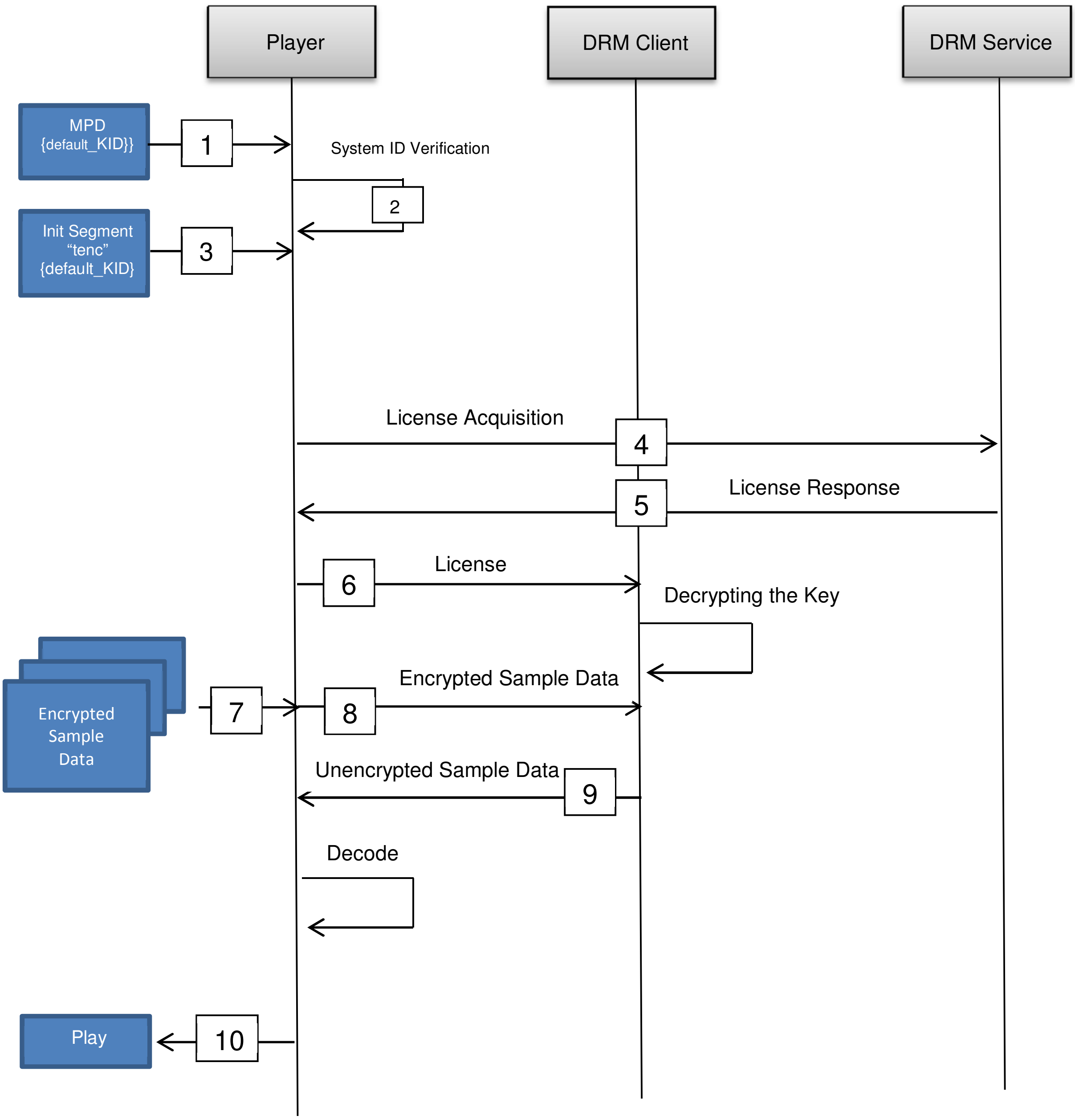
DRM Service – The DRM Provider creates licenses containing a protected media key that can only be decrypted by a trusted client.

The DRM Provider needs to know the default_KID and DRM SystemID and possibly other information like asset ID and player domain ID in order to create and download one or more licenses required for a Presentation on a particular device. Each DRM system has different license acquisition information, a slightly different license acquisition protocol, and a different license format with different playback rules, output rules, revocation and renewal system, etc. The DRM Provider typically must supply the Streamer and the Packager license acquisition information for each UUID ContentProtection Descriptor element or 'pssh' box, respectively.

The DRM Service may also provide logic to manage key rotation, DRM domain management, revocation and renewal and other content protection related features.

Figure 20 below shows a simple workflow with `pssh` information in the Initialization Segment for informational purpose.

Figure 20 Example of Information flow for DRM license retrieval



[1] - A DASH MPD may include the following Content Protection Descriptors to indicate that the 'cenc' scheme (Common Encryption) was used to encrypt the referenced media, and provide license acquisition information for one (or more) DRM system(s) with the indicated SystemID.

[2] – To verify whether specific DRM supported or not using System ID value from MPD.

With unique KIDs, a license request using the `cenc:default_KID` attribute value is sufficient to identify a DRM license containing that key that will enable playback of the Components, Representations, Adaptation Sets, or Periods that the ContentProtection Descriptor element and `default_KID` describe.

[3] - The TrackEncryptionBox (tenc) contains default values for the IsEncrypted flag, IV_size, and KID for the entire track These values are used as the encryption parameters for the samples in this track unless over-ridden by the sample group description associated with a group of samples. The license acquisition information could be also present in PSSH boxes in the initialization segment.

[4] – Decryption Key acquisition. This could be performed either by Player or DRM Client.

[5] – DRM License / Decryption Key response

[6] – License Store. DRM licenses/rights need not be stored in the file in order to look up a key using KID values stored in the file and decrypt media samples using the encryption parameters stored in each track.

[7] – Player requesting encrypted sample data.

[8] – Player provides encrypted sample data to DRM Client for decryption using decryption key. How the DRM system locates the identified decryption key is left to a DRM-specific method

[9] – Player received unencrypted sample data from DRM Client.

7.7. Common Encryption *Test-DRM* Simulation

7.7.1. Introduction

In order to test common encryption without the necessity to do tests for a specific DRM, or all supported DRMs, a common encryption *Test-DRM* simulation is defined.

Specifically the following aspects are defined for the *Test-DRM simulation*:

- To test decryption with common encryption scheme parameters, a clear key and associated KID is provided in a separate file.
- To test the parsing of DRM relevant fields, two different test scenarios are defined to communicate the encryption parameters in the MPD and in the movie. The latter case also includes key rotation.

In the interest of testing independently of a specific DRM system, the keys are provided directly in clear text, in lieu of the DRM specific license information, system keys, and decryption system that is otherwise used to securely obtain the media keys.

The use of an external file allows flexible referencing of the same key from different locations, to e.g. use the same key for audio, video or different Representations.

7.7.2. Test of Common Encryption

The key file location is the MPD directory or configurable in the player to avoid OS dependent path references. Its file name is the KID expressed in 32 hex lower case digits with .txt extension. The content is the decryption key in lower case hex digits e.g.

`bdff1a347bd8e9f523f5ee6b16273d6e.txt` contains:
`050526bf6d3c386ffe5fc17c93506eca`

The key file name can be stored in the `'pssh'` box to verify the creation and parsing of `'pssh'` information. If the `'pssh' box` information is not present, the file name can also be derived directly from the `default_KID` stored in the '`tenc'` box, or a `cenc:default_KID` attribute in the MPD.

In the test vectors 3 different test values for `@schemeIdUri` are defined to represent multi DRMs:

```
00000000-0000-0000-0000-000000000000
00000000-0000-0000-0000-000000000001
00000000-0000-0000-0000-000000000002
```

The test of common decryption is successful when decryption in the above cases is successful.

7.7.3. ContentProtection descriptor

The extension namespace defined in [21] is used to include `default_KID` and `pssh` parameters in the **`ContentProtection`** elements for the test DRM above.

```
 <xs:schema
    targetNamespace="urn:mpeg:cenc:2013"
    attributeFormDefault="unqualified"
    elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:cenc="urn:mpeg:cenc:2013">

    <!-- KID is a 128-bit integer written in canonical UUID string notation -->
    <xs:simpleType name="KeyIdType">
        <xs:restriction base="xs:string">
            <xs:pattern value="[A-Fa-f0-9]{8}-[A-Fa-f0-9]{4}-[A-Fa-f0-9]{4}-[A-Fa-f0-9]{4}-[A-
Fa-f0-9]{12}"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- space-delimited list of KIDs -->
    <xs:simpleType name="KeyIdListType">
        <xs:list itemType="KeyIdType" />
    </xs:simpleType>

    <!-- attribute that can be used within the DASH ContentProtection descriptor -->
    <xs:attribute name="default_KID" type="KeyIdListType"/>

   <!-- element used within system specific UUID ContentProtection descriptors -->

    <xs:element name="pssh" type="xs:base64Binary"/>

</xs:schema>
```

An example is provided below:

```
<ContentProtection schemeIdUri="urn:mpeg:dash:mp4protection:2011"
        value="cenc" cenc:default_KID="34e5db32-8625-47cd-ba06-68fca0655a72"/>

<ContentProtection schemeIdUri="urn:uuid:00000000-0000-0000-0000-000000000000"
        value="DASH264DRM v2.0">
        <cenc:pssh data="cG9zc2libGUgcm9vdCBwc3NoIGxpY2Vuc2UgaW5mbw=="/>
</ContentProtection>
```

Here the `cenc:pssh` element contains the base64 encoded ‘`pssh`’ box and `cenc:default_KID` attribute contains the `default_KID` from the ‘`tenc`’ box, which is the same for all Representations in the Adaptation Set.

7.7.4. Test Scenarios

7.7.4.1. Introduction

Different test scenarios are defined which are then mapped to specific test cases in [27]. The first test scenario uses a single key with

1. `pssh` and `tenc` parameters in the movie box
2. `pssh` element and `tenc.default_KID` parameters in the MPD.

Another test scenario implements key rotation with `tenc` and `pssh` information in the MPD. Finally, a use case for interleaving of unencrypted content is added.

7.7.4.2. Test Scenario 1: pssh and tenc Parameters in Movie Box

The simulation verifies the signaling of the DRM in the MPD, specifically the `pssh` and `tenc` information as it must be exercised to access the keys.

The signaling of encryption scheme(s) in MPD:

```
<ContentProtection schemeIdUri="urn:uuid:00000000-0000-0000-0000-000000000000">
<ContentProtection schemeIdUri="urn:uuid:00000000-0000-0000-0000-000000000001">
<ContentProtection schemeIdUri="urn:uuid:00000000-0000-0000-0000-000000000002">
```

The `pssh` box, if present, contains the base64 encoded filename of the key file.

7.7.4.3. Test Scenario 2: pssh and default_KID Parameters in MPD

The simulation verifies the encoding of the parameters in the MPD. The key file is indicated in the `cenc:pssh` element as base64 encoded KID in lower case with .txt extension. For example, for a KID of `bdff1a347bd8e9f523f5ee6b16273d6e`, the key will be in the file `bdff1a347bd8e9f523f5ee6b16273d6e.txt`.

The `cenc:pssh` element with required base64 encoding in this case is:

```
<cenc:pssh ”YmRmZjFhMzQ3YmQ4ZTlmNTIzZjVlZTZiMTYyNzNkNmUudHh0”/>
```

A separate key file is used for each key when key rotation is used.

7.7.4.4. Test Scenario 3: pssh and KID Parameters in MPD with Key Rotation

In this case, the `pssh` information may contain root license information. For the test scenario, the `pssh` information does not contain relevant key information but is present as a place holder. The static place holder is the base64 encoding of the string: `”possible root pssh license info”`, i.e.:

```
<cenc:pssh data=”cG9zc2libGUgcm9vdCBwc3NoIGxpY2Vuc2UgaW5mbw==”/>
```

A separate key file with different `$KeyId$` value is used for each new key.

7.7.4.5. Test Scenario 4: pssh and tenc Parameters in MPD with Key Rotation and unencrypted elements

This extends the previous test scenario with segments that are signaled as unencrypted that are combined with encrypted segments.

8. DASH-IF Interoperability Points

8.1. Introduction

This version of the document defines Interoperability Points in this section. Earlier versions of this document, especially version 2 [2] defines legacy IOPs.

8.2. DASH-AVC/264 main

8.2.1. Introduction

The scope of the DASH-AVC/264 main interoperability point is basic support of high-quality video distribution over the top based on H.264/AVC up to 1080p. Both, live and on-demand services are supported.

The compliance to DASH-AVC/264 main may be signaled by a `@profiles` attribute with the value "`http://dashif.org/guidelines/dash264main`"

8.2.2. Definition

A DASH client conforms to the IOP by supporting at least the following features:

- All DASH-related features as defined in section 3 of this document.
- The requirements and guidelines in section 4.9.2 for simple live operation.
- The requirements and guidelines in section 5.6.1 for server-based ad insertion.
- H.264/MPEG AVC Progressive High Profile at level 4.0 as defined in section 6.2 together with all AVC-related requirements and recommendation in section 6.2.

- MPEG-4 HE-AAC v2 level 2 profile audio codec as defined in section 6.3. Dynamic Range Control is not expected to be supported.
- subtitle and closed captioning support
    - using SMPTE-TT as defined in section 6.4.2
        - For On-Demand single file download is sufficient.
        - For live services and/or if key rotation is to be supported, the encapsulation into ISO BMFF is necessary.
    - Using CEA-608/708 as defined in section 6.4.3.
- content protection based on common encryption and key rotation as defined in section 7. And specifically, the client supports MPD-based parsing and movie box based parsing of DRM related parameters for common encryption.

Content shall only be authored claiming conformance to this IOP if such a client can properly play the content. In addition, the content shall follow the mandatory aspects and should take into account the recommendations and guidelines for content authoring documented in section 3 (DASH features), section 4.9.2 (simple live operation), section 5.6.1 (server-based ad insertion), AVC-related issues in section 6.2, section 6.3 (audio), section 6.4.2 (SMPTE-TT), section 6.4.3 (CEA-608/708), and section 7 (Content Protection).

If content is offered claiming conformance to this IOP, the content author is encouraged to use the HTTP-URL construction as defined in [7], section 5.1.4.

8.3. DASH-AVC/264 high

8.3.1. Introduction

The scope of the DASH-AVC/264 interoperability point is support of high-quality video distribution over the top based on H.264/AVC up to 1080p. Both, live and on-demand services are supported as well as features for main live and advanced ad insertion.

The compliance to DASH-AVC/264 may be signaled by a `@profiles` attribute with the value `"http://dashif.org/guidelines/dash264high"`

8.3.2. Definition

A client that attempts to consume content generated conforming to this IOP shall support the following features:

- All features required for DASH-264/AVC main as defined in section 8.2.
- The client requirements and recommendations for the main live operation as defined in section 4.9.3.

Content shall only be authored claiming conformance to this IOP if such a client can properly play the content. In addition, the content shall follow the mandatory aspects and should take into account the recommendations and guidelines for content authoring documented in section 8.2 (DASH-264/AVC main), section 4.9.3 (main live operation), and section 5.6.2 (app-based ad insertion).

If content is offered claiming conformance to this IOP, the content author is encouraged to use the HTTP-URL construction as defined in [7], section 5.1.4.

8.4. DASH-IF IOP simple

8.4.1. Introduction

The scope of the DASH-IF IOP simple interoperability point is the basic support of efficient high-quality video distribution over the top with HD video up to 1080p including support for HEVC 8 bit.

The compliance to *DASH-IF IOP simple* may be signaled by a `@profiles` attribute with the value "`http://dashif.org/guidelines/dash-if-simple`"

8.4.2. Definition

A DASH client conforms to the IOP by supporting at least the following features:

- All DASH-related features as defined in section 3 of this document.
- The requirements and guidelines in section 4.9.2 for simple live operation.
- The requirements and guidelines in section 5.6.1 for server-based ad insertion.
- H.264/MPEG AVC Progressive High Profile at level 4.0 as defined in section 6.2 together with all AVC-related requirements and recommendation in section 6.2.
- H.265/MPEG-H HEVC Main Profile Main Tier at level 4.1 as defined in section 6.2 together with all HEVC-related requirements and recommendation in section 6.2.
- MPEG-4 HE-AAC v2 level 2 profile audio codec as defined in section 6.3. Dynamic Range Control is not expected to be supported.
- subtitle and closed captioning support
  - using SMPTE-TT as defined in section 6.4.2
    - For On-Demand single file download is sufficient.
    - For live services and/or if key rotation is to be supported, the encapsulation into ISO BMFF is necessary.
  - Using CEA-608/708 as defined in section 6.4.3.
- content protection based on common encryption and key rotation as defined in section 7. And specifically, the client supports MPD-based parsing and movie box based parsing of DRM related parameters for common encryption.

Content shall only be authored claiming conformance to this IOP if such a client can properly play the content. In addition, the content shall follow the mandatory aspects and should take into account the recommendations and guidelines for content authoring documented in section 3 (DASH features), section 4.9.2 (simple live operation), section 5.6.1 (server-based ad insertion), section 6.2 (video), section 6.3 (audio), section 6.4.2 (SMPTE-TT), section 6.4.3 (CEA-608/708), and section 7 (Content Protection).

If content is offered claiming conformance to this IOP, the content author is encouraged to use the HTTP-URL construction as defined in [7], section 5.1.4.

8.5. DASH-IF IOP Main

8.5.1. Introduction

For the support of broad set of use cases the DASH-IF IOP Main Interoperability Point is defined. In addition the features of DASH-264/AVC main as defined in section 8.2 and DASH-265/HEVC as defined in section 0, the interoperability point requires DASH clients for real-time segment parsing and 10-bit HEVC.

The compliance to *DASH-IF IOP main* may be signalled by a `@profile` attribute with the value `"http://dashif.org/guidelines/dash-if-main"`

8.5.2. Definition

A client that attempts to consume content generated conforming to this IOP shall support the following features:

- All features required for DASH-264/AVC high as defined in section 8.3.
- H.265/MPEG-H HEVC Main Profile Main Tier at level 4.1 as defined in section 6.2 together with all HEVC-related requirements and recommendation in section 6.2.
- H.265/MPEG-H HEVC Main 10 Profile Main Tier at level 4.1 as defined in section 6.2 together with all HEVC-related requirements and recommendation in section 6.2.

Content shall only be authored claiming conformance to this IOP if such a client can properly play the content. In addition, the content shall follow the mandatory aspects and should take into account the recommendations and guidelines for content authoring documented in section 8.3 and HEVC-related issues in section 6.2.

If the content is authored such that it also conforms to DASH-264/AVC high as defined in section 8.3, then the profile identifier for DASH-264/AVC high shall be added as well. If the profile identifier is missing, the content may be considered as HEVC only content.

If content is offered claiming conformance to this IOP, the content author is encouraged to use the HTTP-URL construction as defined in [7], section 5.1.4.

9. Multi-Channel Audio Extension

9.1. Scope

The Scope of the Multichannel Audio Extension is the support of audio with additional channels and codecs beyond the basic audio support as specified in the DASH-AVC/264 base, which is limited to Stereo HE-AAC. Multichannel audio is widely supported in all distribution channels today, including broadcast, optical disc, and digital delivery of audio, including wide support in adaptive streaming delivery.

It is expected that clients may choose which formats (codecs) they support.

9.2. Technologies

9.2.1. Dolby Multichannel Technologies

9.2.1.1. Overview

The considered technologies from Dolby for advanced audio support are:

- Enhanced AC-3 (Dolby Digital Plus) [31]
- Dolby TrueHD [32]

9.2.1.2. DASH-specific issues

In the context of DASH, the following applies:

- The signaling of the different audio codecs for the codecs parameters is documented in [31] and [32], which also provides information on ISO BMFF encapsulation.
- For E-AC-3 the Audio Channel Configuration shall use the " `tag:dolby.com,2014:dash:audio_channel_configuration:2011`" as defined at http://dashif.org/identifiers/audio-source-data/.

Table 19 Dolby Technologies: Codec Parameters and ISO BMFF encapsulation

Codec	Codec Parameter	ISO BMFF Encapsulation	SAP type
Enhanced AC-3 [31]	`ec-3`	ETSI TS 102 366 Annex F [31]	1
Dolby TrueHD	`mlpa`	Dolby [32]	1

9.2.2. DTS-HD

9.2.2.1. Overview

DTS-HD [33] comprises a number of profiles optimized for specific applications. More information about DTS-HD and the DTS-HD profiles can be found at www.dts.com.

9.2.2.2. DASH-specific issues

For all DTS formats SAP is always 1.

The signaling of the various DTS-HD profiles is documented in DTS 9302J81100 [30]. DTS 9302J81100 [30] also provides information on ISO BMFF encapsulation.

Additional information on constraints for seamless switching and signaling DTS audio tracks in the MPD is described in DTS specification 9302K62400 [35].

Table 20: DTS Codec Parameters and ISO BMFF encapsulation

Codec	Codec Parameter	ISO BMFF Encapsulation	SAP type
DT Digital Surround	`dtsc`	DTS 9302J81100 [30]	1

DTS-HD High Resolution and DTS-HD Master Audio	`dtsh`	
DTS Express	`dtse`	
DTS-HD Lossless (no core)	`dtsl`	

9.2.3. MPEG Surround

9.2.3.1. Overview

MPEG Surround, as defined in ISO/IEC 23003-1:2007 [34], is a scheme for coding multichannel signals based on a down-mixed signal of the original multichannel signal, and associated spatial parameters. The down-mix shall be coded with MPEG-4 High Efficiency AAC v2 according to section 5.3.3.

MPEG Surround shall comply with level 4 of the Baseline MPEG Surround profile.

9.2.3.2. DASH-specific issues

In the context of DASH, the following applies for audio codecs

- The signaling of the different audio codecs for the codecs parameters is according to RFC6381 [11] is documented in Table 21. Table 21 also provides information on ISO BMFF encapsulation.
- The content is expected to be prepared according to the MPEG-DASH Implementation Guidelines [7] to make sure each (sub-)segment starts with a SAP of type 1.

Table 21 Codecs parameter according to RFC6381 [11] and ISO BMFF encapsulation for MPEG Surround codec

Codec	Codec Parameter	ISO BMFF Encapsulation	SAP type
MPEG Surround [34]	`mp4a.40.30`	ISO/IEC 14496-14 [8]	1

Note: Since MPEG Surround is based on a down-mix coded with AAC-LC and HE-AAC, for the above mentioned “Codec Parameters” the following is implied:

mp4a.40.30 = AOT 2 + AOT 5 + AOT 30

9.2.4. MPEG-4 High Efficiency AAC Profile v2, level 6

9.2.4.1. Overview

Support for multichannel content is available in the HE-AACv2 Profile, starting with level 4 for 5.1 and level 6 for 7.1. All MPEG-4 HE-AAC multichannel profiles are fully compatible with the DASH-AVC/264 baseline interoperability point for stereo audio, i.e. all multichannel decoders can decode DASH-AVC/264 stereo content.

9.2.4.2. DASH-specific issues

In the context of DASH, the following applies for the High Efficiency AAC v2 Profile

- The content shall be prepared according to the MPEG-DASH Implementation Guidelines [7] to make sure each (sub-)segment starts with a SAP of type 1.
- Signaling of profile levels is not supported in RFC 6381 but the channel configuration shall be signaled by means of the **`ChannelConfiguration`** element in the MPD.
- The signaling of MPEG-4 High Efficiency AAC v2 for the codecs parameters is according to RFC6381 [11] and is documented in Table 22. Table 22 also provides information on the ISO BMFF encapsulation.
- For all HE-AAC bitstreams, explicit backward-compatible signaling of SBR shall be used.
- The content should be prepared incorporating loudness and dynamic range information into the bitstream also considering DRC Presentation Mode in ISO/IEC 14496-3 [12], Amd. 4.
- Decoders shall support decoding of loudness and dynamic range related information, i.e. dynamic_range_info() and MPEG4_ancillary_data() in the bitstream.

Table 22 Codecs parameter according to RFC6381 [11] and ISO BMFF encapsulation

Codec	Codec Parameter	ISO BMFF Encapsulation	SAP type
MPEG-4 AAC Profile [12]	`mp4a.40.2`	ISO/IEC 14496-14 [13]	1
MPEG-4 HE-AAC Profile [12]	`mp4a.40.5`	ISO/IEC 14496-14 [13]	1
MPEG-4 HE-AAC v2 Profile [12]	`mp4a.40.29`	ISO/IEC 14496-14 [13]	1

Note: Since both, HE-AAC and HE-AACv2 are based on AAC-LC, for the above mentioned "Codec Parameters" the following is implied:

mp4a.40.5 = AOT 2 + AOT 5

9.3. Client Implementation Guidelines

Independent of the codec, a client that supports one or more codecs of multichannel sound playback should exhibit the following characteristics:

- Playback multichannel sound correctly given the client operating environment. As an example, if the audio track delivers 5.1 multichannel sound, the client might perform one or more of the following: decode the multichannel signal on the device and output either 6ch PCM over HDMI, or pass that multichannel audio with no changes to external AVRs, or if the device is rendering to stereo outputs such as headphones, either correctly downmix that multi-channel audio to 2-channel sound, or select an alternate stereo adaptation set, or other appropriate choices.

- Adaptively and seamless switch between different bitrates as specified in the adaptation sets according to the playback clients logic. Seamless switching is defined as no perceptible interruption in the audio, and no loss of A/V sync. There is no expectation that a client can seamlessly switch between formats.

9.4. Extensions

9.4.1. General

9.4.1.1. Definitions

A *multichannel audio client* at least supports the following features:

- All DASH-related features as defined in section 3 of this document.
- content protection based on common encryption and key rotation as defined in section 7. And specifically, the client supports MPD-based parsing and movie box based parsing of DRM related parameters for common encryption.
- The client implementation guidelines in section 9.3.

9.4.1.2. Recommendations

If content is offered claiming conformance to any extension in this section, the content author is encouraged to use the HTTP-URL construction as defined in [7], section 5.1.4.

9.4.2. Dolby Extensions

9.4.2.1. Introduction

For the support of Dolby advanced audio support, two additional extensions are defined.

Compliance to *DASH-IF multichannel audio extension with Enhanced AC-3* (Dolby Digital Plus) [31] may be signaled by an `@profile` attribute with the value `"http://dashif.org/guidelines/dashif#ec-3"`.

Compliance to *DASH-IF multichannel extension with Dolby TrueHD* may be signaled by an `@profile` attribute with the value `"http://dashif.org/guidelines/dashif#mlpa"`.

9.4.2.2. Supporters

These extensions are supported by the following DASH IF members: Dolby, DTS, Fraunhofer, BuyDRM, Sony.

9.4.2.3. Definition

Content may be authored claiming conformance to *DASH-IF multichannel audio extension with Enhanced AC-3*

- if the content is multichannel audio content as defined in section 9.4.1, and
- if a client can properly play the content by supporting at least the following features
  - all multichannel audio client features as defined in section 9.4.1

- Enhanced AC-3 (Dolby Digital Plus) [31] and the DASH-specific features defined in section 9.2.1.2

Content may be authored claiming conformance to *DASH-IF multichannel extension with Dolby TrueHD*

- if the content is multichannel audio content as defined in section 9.4.1, and
- if a client can be properly play the content by supporting at least the following features
  - all multichannel audio client features as defined in section 9.4.1
  - Dolby TrueHD and the DASH-specific features defined in section 9.2.1.2

9.4.3. DTS-HD Interoperability Points

9.4.3.1. Introduction

For the support of DTS advanced audio support, four additional extensions are defined.

Compliance to *DASH-IF multichannel audio extension with DTS Digital Surround* may be signaled by a `@profile` attribute with value `"http://dashif.org/guidelines/dashif#dtsc"`.

Compliance to *DASH-IF multichannel audio extension with DTS-HD High Resolution and DTS-HD Master Audio* may be signaled by a `@profile` attribute with value `"http://dashif.org/guidelines/dashif#dtsh"`

Compliance to *DASH-IF multichannel audio extension with DTS Express* may be signaled by a `@profile` attribute with value `"http://dashif.org/guidelines/dashif#dtse"`

Compliance to *DASH-IF multichannel extension with DTS-HD Lossless (no core)* may be signaled by a `@profile` attribute with value `"http://dashif.org/guidelines/dashif#dtsl"`

9.4.3.2. Supporters

These extensions are supported by the following DASH IF members: Dolby, DTS, Fraunhofer, BuyDRM, Sony.

9.4.3.3. Definition

Content may be authored claiming conformance to *DASH-IF multichannel audio extension with DTS Digital Surround*

- if the content is multichannel audio content as defined in section 9.4.1, and
- if a client can be properly play the content by supporting at least the following features
  - all multichannel audio client features as defined in section 9.4.1
  - DTS and the DASH-specific features defined in section 9.2.2.2

Content may be authored claiming conformance to *DASH-IF multichannel audio extension with DTS-HD High Resolution and DTS-HD Master Audio*

- if the content is multichannel audio content as defined in section 9.4.1, and

- if a client can be properly play the content by supporting at least the following features
    - all multichannel audio client features as defined in section 9.4.1
    - DTS-HD High Resolution and DTS-HD Master Audio and the DASH-specific features defined in section 9.2.2.2

Content may be authored claiming conformance to *DASH-IF multichannel audio extension with DTS Express*

- if the content is multichannel audio content as defined in section 9.4.1, and
- if a client can be properly play the content by supporting at least the following features
    - all multichannel audio client features as defined in section 9.4.1
    - DTS-HD Express and the DASH-specific features defined in section 9.2.2.2

Content may be authored claiming conformance to *DASH-IF multichannel extension with DTS-HD Lossless (no core)*

- if the content is multichannel audio content as defined in section 9.4.1, and
- if a client can be properly play the content by supporting at least the following features
    - all multichannel audio client features as defined in section 9.4.1
    - DTS-HD Lossless (no core) and the DASH-specific features defined in section 9.2.2.2

9.4.4. MPEG Surround Interoperability Points

9.4.4.1. Introduction

For the support of MPEG Surround advanced audio support the following extension is defined.

Compliance to *DASH-IF multichannel audio extension with MPEG Surround* according to ISO/IEC 23003-1:2007 [34] may be signaled by an `@profile` attribute with the value `"http://dashif.org/guidelines/dashif#mps"`.

9.4.4.2. Supporters

These extensions are supported by the following DASH IF members: Dolby, DTS, Fraunhofer, BuyDRM, Sony.

9.4.4.3. Definition

Content may be authored claiming conformance to *DASH-IF multichannel audio extension with MPEG Surround*

- if the content is multichannel audio content as defined in section 9.4.1, and
- if a client can be properly play the content by supporting at least the following features
    - all multichannel audio client features as defined in section 9.4.1
    - ISO/IEC 23003-1:2007 and the DASH-specific features defined in section 9.2.3.2

9.4.5. MPEG HE-AAC Multichannel Interoperability Points

9.4.5.1. Introduction

Compliance to *DASH-IF multichannel audio extension with HE-AACv2 level 4* [12] may be signaled by an @profile attribute with the value `"http://dashif.org/guidelines/dashif#heaac-mc51"`.

Compliance to *DASH-IF multichannel audio extension with HE-AACv2 level 6* [12] may be signaled by an @profile attribute with the value `"http://dashif.org/guidelines/dashif#heaac-mc71"`.

9.4.5.2. Supporters

These extensions are supported by the following DASH IF members: Dolby, DTS, Fraunhofer, BuyDRM, Sony.

9.4.5.3. Definition

Content may be authored claiming conformance to *DASH-IF multichannel audio extension with HE-AACv2 level 4*

- if the content is multichannel audio content as defined in section 9.4.1, and
- if a client can be properly play the content by supporting at least the following features
  - all multichannel audio client features as defined in section 9.4.1
  - HE-AACv2 level 4 [12] and the DASH-specific features defined in section 9.2.4.2

Content may be authored claiming conformance to *DASH-IF multichannel audio extension with HE-AACv2 level 6*

- if the content is multichannel audio content as defined in section 9.4.1, and
- if a client can be properly play the content by supporting at least the following features
  - all multichannel audio client features as defined in section 9.4.1
  - HE-AACv2 level 6 [12] and the DASH-specific features defined in section 9.2.4.2

Annex A Examples for Profile Signalling

Example 1

In this case DASH-AVC/264 content is offered, but in addition a non-conforming Adaptation Set is added.

Here is an example for an MPD:

- **MPD**@profiles="urn:mpeg:dash:profile:isoff-on-demand:2011, http://dashif.org/guidelines/dash264"
    - **AdaptationSet**@profiles="urn:mpeg:dash:profile:isoff-on-demand:2011, http://dashif.org/guidelines/dash264"
    - **AdaptationSet**@profiles ="http://dashif.org/guidelines/dash264"
    - **AdaptationSet**@profiles ="urn:mpeg:dash:profile:isoff-on-demand:2011"

Pruning process for IOP `http://dashif.org/guidelines/dash264` results in

- **MPD**@profiles ="http://dashif.org/guidelines/dash264"
    - **AdaptationSet**@profiles ="http://dashif.org/guidelines/dash264"
    - **AdaptationSet**@profiles ="http://dashif.org/guidelines/dash264"

It is now required that the pruned MPD conforms to DASH-AVC/264.

Example 2

In this case DASH-AVC/264 content is offered, but in addition a non-conforming Adaptation Set is added and one DASH-IF Example Extension Adaptation Set is added with the virtual IOP signal `http://dashif.org/guidelines/dashif#extension-example`.

Here is an example for an MPD:

- **MPD**@profiles ="urn:mpeg:dash:profile:isoff-on-demand:2011, http://dashif.org/guidelines/dash264, http://dashif.org/guidelines/dashif#extension-example"
    - @id = 1, **AdaptationSet**@profiles ="urn:mpeg:dash:profile:isoff-on-demand:2011, http://dashif.org/guidelines/dash264"
    - @id = 2, **AdaptationSet**@profiles ="http://dashif.org/guidelines/dash264"
    - @id = 3, **AdaptationSet**@profiles ="urn:mpeg:dash:profile:isoff-on-demand:2011, http://dashif.org/guidelines/dashif#extension-example"

Pruning process for profile `http://dashif.org/guidelines/dash264` results in

- MPD@profiles="http://dashif.org/guidelines/dash264"
    - @id = 1, **AdaptationSet**@profiles="http://dashif.org/guidelines/dash264"
    - @id = 2, **AdaptationSet**@profiles="http://dashif.org/guidelines/dash264"

It is now required that the pruned MPD conforms to DASH-AVC/264.

Pruning process for profile `http://dashif.org/guidelines/dashif#extension-example` results in

- **MPD**@profiles="http://dashif.org/guidelines/dash264"
    - @id = 3, **AdaptationSet**@profiles="http://dashif.org/guidelines/dashif# extension-example"

It is now required that the pruned MPD conforms to DASH-IF Example Extension Adaptation Set.

Annex B Live Services - Use Cases and Architecture

B.1 Baseline Use cases

B.1.1 Use Case 1: Live Content Offered as On-Demand

In this case content that was distributed as live is offered in a separate Media Presentation as On-Demand Content.

B.1.2 Use Case 2: Scheduled Service with known duration and Operating at live edge

In this case a service started a few minutes ago and lasts 30 minutes. The duration is known exactly and also all segment URLs are known. The timeshift buffer is short. This may for example be a live service for which the service provider wants to ensure that only a small window is accessible. The content is typically be pre-canned, but offered in a scheduled manner.

B.1.3 Use Case 3: Scheduled Service with known duration and Operating at live edge and time shift buffer

In this case a service started a few minutes ago and lasts 30 minutes. The duration is known exactly and also all segment URLs are known. The timeshift buffer is long. This may for example be a service for which the service provider wants to ensure that the content is made available in a scheduled manner, e.g. no client can access the content earlier than scheduled by the content provider. However, after the live edge is completed, the content is available for 24h. The content is typically pre-canned.

B.1.4 Use Case 4: Scheduled Live Service known duration, but unknown Segment URLs

In this case a live service started a few minutes ago and lasts 30 minutes. The duration is known exactly but the segment URLs are unknown, as for example some advertisement may be added on the fly. Otherwise this service is similar to use case 3.

B.1.5 Use Case 5: 24/7 Live Service

In this case a live service started that may have started a long time ago is made available. Ad breaks and operational updates may be done with a 30sec pre-warning. The duration is unknown and also the segment URLs, the exact set of provided media components (different language tracks, subtitles, etc.) are unknown, as for example some advertisement may be added on the fly. Otherwise this service is similar to use case 3.

B.1.6 Use Case 6: Approximate Media Presentation Duration Known

In this case a live service starts at a specific time. The duration is known approximately and also all segment URLs are known for the approximate duration. Towards the end of the Media Presentation, the Media Presentation duration may be extended or may be finally determined by providing an update of the MPD.

B.2 Baseline Architecture for DASH-based Live Service

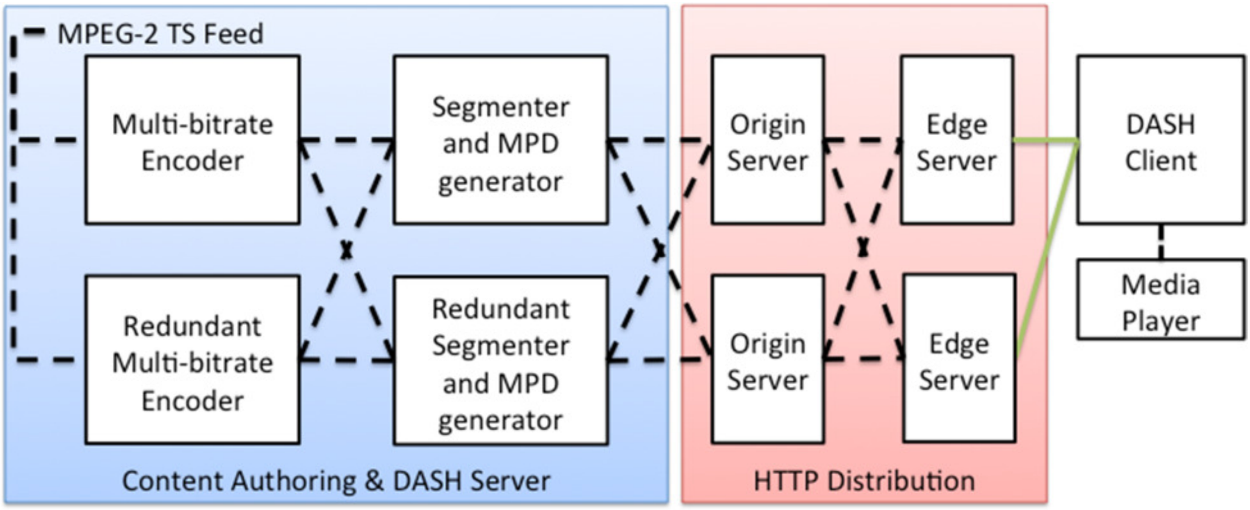


Figure 21 Typical Deployment Scenario for DASH-based live services

The figure depicts a redundant set-up for Live DASH with unicast. Function redundancy is added to mitigate the impact of function failures. The redundant functions are typically connected to multiple downstream functions to mitigate link failure impacts.

An MPEG2-TS stream is used often as feed into the encoder chain. The multi-bitrate encoder produces the required number of Representations for each media component and offers those in one Adaptation Set. In the context of this document is assumed that content is offered in the ISO BMFF live profile with the constraints according to v2 of this document. The encoder typically locks to the system clock from the MPEG2-TS stream. The encoder forwards the content to the segmenter, which produces the actual DASH segments and handles MPD generation and updates. Content Delivery Network (CDN) technologies are typically used to replicate the content to multiple edge servers. Note: the CDN may include additional caching hierarchy layers, which are not depicted here.

Clients fetch the content from edge servers using HTTP (green connection) according to the MPEG-DASH and DASH-IF IOP specification. Different protocols and delivery formats may be used within the CDN to carry the DASH segments from the segmenter to the Edge Server. For instance, the edge server may use HTTP to check with its parent server when a segment is not (yet) in the local cache. Or, segments may be pushed using IP Multicast from the origin server to relevant edge servers. Other realizations are possible, but are outside of the normative scope of this document.

In some deployments, the live service is augmented with ad insertion. In this case, content may not be generated continuously, but may be interrupted by ads. Ads itself may be personalized, targeted or regionalized.

B.3 Distribution over Multicast

This section describes a baseline architecture for DASH Live Services for broadcast distribution. The intention of the baseline architecture is in particular to identify robustness and failure issue and give guidance on procedures to recover.

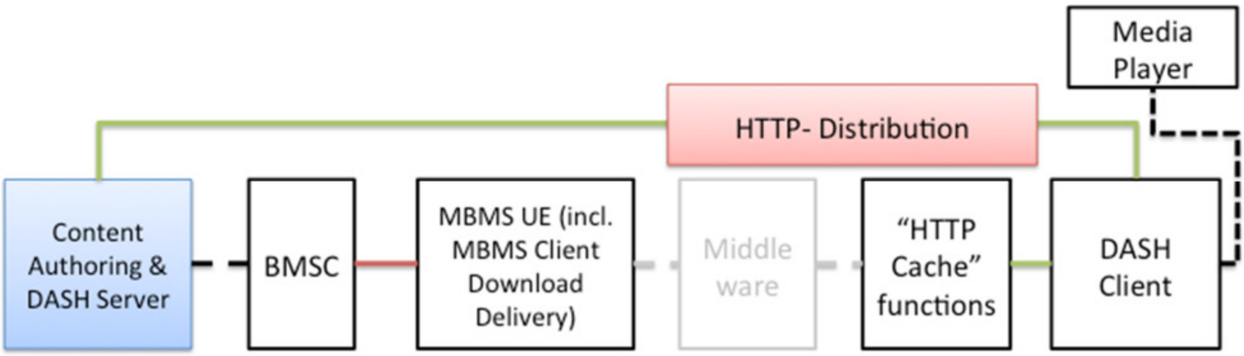


Figure 22 Typical Deployment Scenario for DASH-based live services partially offered through MBMS (unidirectional FLUTE distribution)

The same content authoring and DASH server solution as shown in Figure 1 are considered in this baseline architecture. The DASH Segmenter (cf. Fig .1) provides DASH segments of typically one quality representation into the BM-SC, which sends the segments using MBMS Download (as sequence of files using IETF FLUTE protocol) to the MBMS User Equipment (UE). The MBMS UE includes the needed MBMS download delivery client functions to recover the media segments from the FLUTE reception. The MBMS UE makes the segments through a local HTTP Cache function available to the DASH client. The DASH client uses HTTP (green line) to retrieve the segments from the device local cache.

In case the MBMS reception is not possible for that Video Session, the DASH client can use unicast HTTP to acquire the stream (according to previous section).

Note, the objective of the client architecture realization here is on using a generic DASH client for unicast and broadcast. More customized implementations are possible.

B.4 Typical Problems in Live Distribution

B.4.1 Introduction

Based on the deployment architectures in Figure 21 and Figure 22 a few typical problems in DASH-based ABR distribution are explained.

B.4.2 Client Server Synchronization Issues

In order to access the DASH segments at the proper time as announced by the segment availability times in the MPD, client and server need to operate in the same time source, in general a globally accurate wall-clock, for example provided by NTP or GPS. There are different reasons why the DASH client and the media generation source may not have identical time source, such as

- DASH client is off because it does not have any protocol access to accurate timing. This may for example be the case for DASH clients that are running in the browser or on top of a general-purpose HTTP stack according to RFC 2616 [22].
- DASH client clock drifts against the system clock and the DASH client is not synchronizing frequently enough against the time-source.
- The segmenter synchronized against a different time source than DASH client.
- There may be unknown delay on the ingest to the server/cache whether the segment is accessible. This is specifically relevant if MBMS is used as the contribution link resulting in transport delay.

- It may also be that the MPD provides the availability times at the segmenter, but the actual availability should be the one on the origin server.
- There may be a delay from segmenter to the origin server which is known by edge/origin, but there may not be sufficient ways to signal this delay.

B.4.3 Synchronization Loss of Segmenter

The segmenter as depicted in Figure 21 may lose synchronization against the input timeline for reasons such as power-outage, cord cuts, CRC losses in the incoming signals, etc. In this case:

- Loss of synchronization may result that the amount of lost media data cannot be predicted which makes the generation of continuous segments difficult.
- The Segmenter cannot predict and correct the segment timeline based on media presentation timestamps, since the presentation timeline may contain a discontinuity due to the synchronization loss
    - a loss of sync (e.g. CRC failure on the input stream)
    - a power glitch on the source
    - someone pulling a cable
- There are cases where no media segments are available, but the MPD author knows this and just wants to communicate this to the receiver.

B.4.4 Encoder Clock Drift

In certain cases, the MBR encoder is slaved to the incoming MPEG-2 TS, i.e. it reuses the media time stamps also for the ISO BMFF.

- What may occur that the encoder clock drifts between the sender and the receivers (longer term issue) , e.g. due to encoder clock tolerance
    - Example: Encoder produces frame every 39.97ms instead of 40ms
    - Tolerance in MPEG-2 TS: 1 frame every 18 minutes
- This may create issues in particular when an existing stream like for satellite is transcoded and segmented into DASH representations.
- Annex A.8 of ISO 23009-1 handles drift control of the media timeline, but the impact on the segment availability time (i.e. MPD updates) is not considered or suggested.
- In particular when the segment fetching engine of the client is only working with the segment availability timeline (so is not parsing the presentation timeline out of the segments), the segment fetching engine will not fetch the segments with the correct interval, leading to buffer underruns or increased e2e delay.
- There is practical evidence that this is a problem in actual deployments, may result in drifts of minutes over hours.

B.4.5 Segment Unavailability

When a server cannot serve a requested segment it gives an HTTP 404 response. If the segment URL is calculated according to the information given in the MPD, the client can often interpret the 404 response as a possible synchronization issue, i.e. its time is not synchronized to the time offered in the MPD.

In the MBMS case, a 404 response is also likely to be caused by non-reparable transport errors. This is even more likely if it has been possible to fetch segments according to the MPD information earlier. Although the client M/W, which is normally located in the same device as the DASH player, knows what segments have been delivered via broadcast and which ones are missing in a sequence, it cannot indicate this to the DASH client using standard HTTP responses to requests for media segments.

B.4.6 Swapping across Redundant Tools

In case of failures, redundant tools kick in. If the state is not fully maintained across redundant tools, the service may not be perceived continuous by DASH client. Problems that may happen at the encoder, that redundant encoders do not share the same timeline or the timeline is interrupted. Depending on the swap strategy ("hot" or "warm"), the interruptions are more or less obvious to the client. Similar issues may happen if segmenters fail, for example the state for segment numbering is lost.

B.4.7 CDN Issues

Typical CDN operational issues are the following:

- Cache Poisoning – at times segment generation may be erroneous. The encoder can produce a corrupt segment, or the segment can become corrupted during upload to origin. This can happen for example if encoder connectivity fails in mid segment upload, leading to a malformed segment (with the correct name) being sent to edge and caching servers. The CDN then caches this corrupt segment and continues to deliver it to fulfill future requests, leading to widespread client failures.
- Cache inconsistency – with a dual origin scheme, identically named segments can be produced with slight differences in media time, due to clock drift or other encoder issues. These segments are then cached by CDNs and used to respond to client requests. If segments from one encoder are mixed with segments of another, it can lead to discontinuous playback experiences on the clients.

B.4.8 High End-to-end Latency

End-to-end latency (also known as hand-waving latency) is defined as the accumulated delay between an action occurring in front of the camera and that action being visible in a buffered player. It is the sum of

1. Encoder delay in generating a segment.
2. Segment upload time to origin server from the encoder.
3. Edge server segment retrieval time from origin
4. Segment retrieval time by the player from the edge server
5. The distance back from the live point at which the player chooses to start playback.
6. Buffering time on the player before playback commences.

In steps 1 through 4, assuming non-chunked HTTP transfer, the delay is a linear function of the segment duration. Overly conservative player buffering can also introduce unnecessary delay, as can choosing a starting point behind the live point. Generally the further behind live the player chooses to play, the more stable the delivery system is, which leads to antagonistic demands on any production system of low latency and stability.

B.4.9 Buffer Management & Bandwidth Estimation

The main user experience degradations in video streaming are rebuffering events. At the same time, user experience is influenced by the quality of the video (typically determined by the bitrate) as well as at least for certain cases on the end-to-end latency. In order to request the access bitrate, the client does a bandwidth estimation typically based on the history and based on this and the buffer level in the client it decides to maintain or switch Representations.

In order to compensate bandwidth variations, the client buffers some media data prior to play-out. More time buffer results less buffer under runs and less rebuffering, but increases end-to-end latency. In order to maximize the buffer in the client and minimize the end-to-end latency the DASH client would like to request the media segment as close as possible to its actual segment availability start time. However, this may cause issues in the playout as the in case of bitrate variations, the buffer may drain quickly and result in playout starvation and rebuffering.

B.4.10 Start-up Delay and Synchronization Audio/Video

At the start-up and joining, it is relevant that the media playout is initiated, but that the delay at start is reasonable and that the presentation is enabled such that audio and video are presented synchronously. As audio and video Representations typically are offered in different sampling rates, and segments of audio and video are not aligned at segment boundaries. Hence, for proper presentation at startup, it is necessary that the DASH client schedules the presentation at the presentation time aligned to the over media presentation timeline.

B.5 Advanced Use Cases

B.5.1 Introduction

Based on the above issues a few advanced use cases are considered.

B.5.2 Use Case 7: Live Service with undetermined end

In this case a live service started that may have started a long time ago is made available. The MPD update may be done with a 30sec pre-warning. The duration is unknown exactly but the segment URLs are unknown, as for example some advertisement may be added on the fly. Otherwise this service is similar to use case 3.

B.5.3 Use Case 8: 24/7 Live Service with canned advertisement

In this case a live service started that may have started a long time ago is made available. The MPD update may be done with a 30sec pre-warning. The duration is unknown exactly but the segment URLs are unknown, as for example some advertisement may be added on the fly. The advertisement itself is not a dynamic service, but available on a server as a pre-canned advertisement.

B.5.4 Use case 9: 24x7 live broadcast with media time discontinuities

In other use cases, the content provider splices content such as programs and ads with independent media timelines at the content provider.

B.5.5 Use case 10: 24x7 live broadcast with Segment discontinuities

Based on the discussions above, interruptions in encoding, etc., but presentation and media timelines resume after loss of some segments.