

Guidelines for Implementation: DASH-IF Interoperability Points

Living Document, 25 September 2019

This version:

<https://dashif.org/guidelines/>

Issue Tracking:

[GitHub](#)

[Inline In Spec](#)

Editors:

DASH Industry Forum

Table of Contents

1	Purpose
2	Interpretation
3	Disclaimer
4	DASH and related standards
4.1	Relationship to the previous versions of this document
4.2	Structure of a DASH presentation
5	Interoperability requirements
5.1	CMAF and ISO BMFF Requirements
5.2	Timing model
5.2.1	Conformance requirements
5.2.2	MPD Timeline
5.2.3	Periods
5.2.4	Representations
5.2.5	Sample timeline
5.2.6	Clock drift is forbidden
5.2.7	Media segments
5.2.7.1	Media segment duration deviation
5.2.7.2	Segments must be aligned
5.2.8	Period connectivity
5.2.8.1	Period continuity
5.2.9	Dynamic MPDs
5.2.9.1	Real time clock synchronization
5.2.9.2	Availability
5.2.9.3	Time shift buffer
5.2.9.4	Presentation delay
5.2.9.5	MPD updates
5.2.9.5.1	Adding content to the MPD
5.2.9.5.2	Removing content from the MPD
5.2.9.5.3	End of live content
5.2.9.6	MPD refreshes
5.2.10	Timing of stand-alone IMSC1 and WebVTT text files
5.2.11	Forbidden techniques
5.2.12	Examples
5.2.12.1	Offer content with imperfectly aligned tracks
5.2.12.2	Split a period
5.2.12.3	Change the default_KID

5.3	Segment addressing modes
5.3.1	Indexed addressing
5.3.2	Structure of the index segment
5.3.2.1	Moving the period start point (indexed addressing)
5.3.3	Explicit addressing
5.3.3.1	Moving the period start point (explicit addressing)
5.3.4	Simple addressing
5.3.4.1	Inaccuracy in media segment timing when using simple addressing
5.3.4.2	Moving the period start point (simple addressing)
5.3.4.3	Converting simple addressing to explicit addressing
5.4	Adaptation set contents
5.5	Adaptation set types
5.6	Video adaptation set constraints
5.7	Audio adaptation set constraints
5.8	Text adaptation set constraints
5.9	Accessing resources over HTTP
5.9.1	MPD URL resolution
5.9.2	Segment URL resolution
5.9.3	Conditional MPD downloads
5.9.4	Expanding URL template variables
5.10	Minimum buffer time signaling
5.11	Large timescales and time values
5.12	MPD size
5.13	Representing durations in XML
6	Commonly used features
6.1	Seamless switching
6.2	Preview thumbnails for seeking and navigation
6.3	Trick mode
6.4	Bitstream switching
6.5	Switching across adaptation sets
6.6	XLink
6.7	Update signaling via in-band events
6.8	Specifying initial position in presentation URL
7	Content annotation and selection
7.1	Annotations for content selection
7.2	Content model
7.2.1	Signaling alternative content
7.2.2	Signaling associated content
7.3	Client processing reference model
8	On-demand services
8.1	Surviving transforming boxes and other adaptation middleboxes
9	Live services
9.1	Selecting the time shift buffer size
9.2	Selecting the suggested presentation delay
9.3	Selecting the media segment duration
9.4	Safety margins in availability timing
9.5	Selecting the minimum update period
9.6	Robust and seamless period transitions
9.7	Determining the live edge
9.8	Trick mode for live services
9.9	DVB-DASH alignment
9.10	Converting a live service to an on-demand service
9.11	Reliable and consistent-delay live service
9.11.1	Consistent latency
9.11.2	Unanticipated new periods
9.11.3	Media segment duration variations
9.11.4	Losses and operational failures
9.11.5	Minimizing MPD updates
9.11.6	Proposed service configuration and MPD generation logic

- 9.11.6.1 Service configuration for simple live
- 9.11.6.2 Service configuration for main live

10 Ad insertion

- 10.1 Remote elements
- 10.2 Periods
- 10.3 Segment availability
- 10.4 Seamless transition
- 10.5 Period labeling
- 10.6 DASH events
- 10.7 MPD updates
- 10.8 Session information
- 10.9 Tracking and reporting
- 10.10 Ad insertion architectures
- 10.11 Server-based architecture
 - 10.11.1 Implementation basics
 - 10.11.2 Remote period elements
 - 10.11.3 Timing and dereferencing
 - 10.11.4 Asset identifiers
 - 10.11.5 MPD updates
 - 10.11.6 MPD events
 - 10.11.7 Workflows
 - 10.11.8 Linear workflow
 - 10.11.8.1 Cue interpretation by the MPD generator
 - 10.11.8.2 Cue interpretation by the packager
 - 10.11.8.3 Multiple cue messages
 - 10.11.8.4 Cancellation
 - 10.11.8.5 Early termination
 - 10.11.8.6 Informational cue messages
 - 10.11.8.7 Ad decision
 - 10.11.9 On demand workflow
 - 10.11.9.1 Capture to VoD
 - 10.11.9.2 Slates and ad replacement
 - 10.11.9.3 Blackouts and alternative content
 - 10.11.9.4 Tracking and reporting
 - 10.11.10 Examples
 - 10.11.11 Use of query parameters
- 10.12 App-based architecture
 - 10.12.1 Implementation basics
 - 10.12.2 SCTE 35 events
 - 10.12.3 Asset identifiers
 - 10.12.4 Linear workflow
 - 10.12.5 On demand workflow
- 10.13 AssetIdentifier extensions
- 10.14 Remote period extensions
- 10.15 User-defined event extensions
 - 10.15.1 Cue message
 - 10.15.2 Reporting
 - 10.15.3 Ad insertion event streams

11 Media coding technologies

- 11.1 H.264 (AVC)
- 11.2 H.265 (HEVC)
- 11.3 Decoder configuration with H.264 and H.265
- 11.4 Bitstream switching with H.264 and H.265
- 11.5 Thumbnail images
- 11.6 HE-AACv2 audio (stereo)
- 11.7 HE-AACv2 audio (multichannel)
- 11.8 CEA-608/708 Digital Television (DTV) Closed Captioning
- 11.9 Timed Text (IMSC1)
- 11.10 Enhanced AC-3 (Dolby Digital Plus)
- 11.11 Dolby TrueHD
- 11.12 AC-4

- 11.13 DTS-HD
- 11.14 MPEG Surround
- 11.15 MPEG-H 3D Audio
- 11.16 MPEG-D Unified Speech and Audio Coding
- 11.17 UHD HEVC 4K
 - 11.17.1 TS 103.433 HDR dynamic metadata
 - 11.17.2 HEVC UHD compatibility aspects
- 11.18 HEVC HDR PQ10
 - 11.18.1 HEVC PQ10 HDR dynamic metadata
 - 11.18.2 SMPTE 2094-10 HDR dynamic metadata
 - 11.18.3 SMPTE 2094-40 HDR dynamic metadata
- 11.19 UHD Dual-Stream (Dolby Vision)
 - 11.19.1 Requirements for enhancement layer
- 11.20 VP9
 - 11.20.1 HD
 - 11.20.2 UHD
 - 11.20.3 HDR

12 Content protection and security

- 12.1 Introduction
- 12.2 HTTPS and DASH
- 12.3 Content Encryption
- 12.4 ISOBMFF Support for Common Encryption and DRM
 - 12.4.1 ISOBMFF Structure Overview
 - 12.4.2 ISOBMFF Content Protection Constraints
- 12.5 DASH MPD Support for Common Encryption and DRM
 - 12.5.1 MPD Structure Overview
 - 12.5.1.1 `ContentProtection` Element for the `mp4protection` Scheme
 - 12.5.1.2 `ContentProtection` Element for the UUID Scheme
 - 12.5.1.3 `cenc`: Namespace Extension
 - 12.5.2 MPD Content Protections Constraints
- 12.6 Mix ISOBMFF and MPD Content Protections Constraints
- 12.7 Client Interactions with DRM Systems
- 12.8 Additional Constraints for Specific Use Cases
 - 12.8.1 Periodic Re-Authorization
 - 12.8.1.1 Periodic Re-Authorization Content Protections Constraints
 - 12.8.1.2 Implementation Options
 - 12.8.2 Low Latency
 - 12.8.2.1 Licenses Pre-Delivery
 - 12.8.2.2 Key Hierarchy and CMAF Chunked Content
 - 12.8.3 Use of W3C Clear Key with DASH
 - 12.8.4 License Acquisition URL XML Element `Laurl`
 - 12.8.4.1 ClearKey Example Using `Laurl`

13 Annex B

14 Annex: Dolby Vision streams within ISO BMFF

15 Annex: Signaling Dolby Vision profiles and levels

16 Annex: Display management message

17 Annex: Composing metadata message

18 Annex: Sample Dual-layer MPD

19 Externally defined terms

Conformance

Index

Terms defined by this specification

References

Normative References
Informative References

Issues Index

1. Purpose§

The scope of the DASH-IF InterOperability Points (**IOPs**) defined in this document is to provide support interoperable services for high-quality video distribution based on MPEG-DASH and related standards. The specified features enable relevant use cases including on-demand and live services, ad insertion, content protection and subtitling. The integration of different media codecs into DASH-based distribution is also defined.

The guidelines are provided in order to address DASH-IF members' needs and industry best practices. The guidelines provide support the implementation of conforming service offerings as well as the DASH client implementation. While alternative interpretations may be equally valid in terms of standards conformance, services and clients created following the guidelines defined in this document can be expected to exhibit highly interoperable behavior between different implementations.

Any identified bugs or missing features may be submitted through the DASH-IF issue tracker at <https://gitreports.com/issue/Dash-Industry-Forum/DASH-IF-IOP>.

2. Interpretation§

Requirements in this document describe required service and client behaviors that DASH-IF considers interoperable:

1. If a service provider follows these requirements in a published DASH service, that service is likely to experience successful playback on a wide variety of clients and exhibit graceful degradation when a client does not support all features used by the service.
2. If a client implementer follows the client-oriented requirements described in this document, the client plays the content conforming to this document.

This document uses statements of fact when describing normative requirements defined in referenced specifications such as [\[MPEGDASH\]](#) and [\[MPEGCMAF\]](#). [\[\[RFC2119!\]\]](#) statements (e.g. "SHALL", "SHOULD" and "MAY") are used when this document defines a new requirement or further constrains a requirement from a referenced document. In order to clearly separate the requirements of referenced specifications vs. the additional requirements set by this document, the normative statements in each section of this document are separated into two different groups, ones starting with "(referenced specification) requires/recommends:" and the ones starting with "This document requires/recommends:". See also [Conformance](#).

All DASH presentations are assumed to be conforming to an [IOP](#). A service may explicitly signal itself as conforming by including the string <https://dashif.org/guidelines/> in [MPD@profiles](#).

There is no strict backward compatibility with previous versions - best practices change over time and what was once considered sensible may be replaced by a superior approach later on. Therefore, clients and services that were conforming to version N of this document are not guaranteed to conform to version N+1.

3. Disclaimer§

This is a document made available by DASH-IF. The technology embodied in this document may involve the use of intellectual property rights, including patents and patent applications owned or controlled by any of the authors or developers of this document. No patent license, either implied or express, is granted to you by this document. DASH-IF has made no search or investigation for such rights and DASH-IF disclaims any duty to do so. The rights and obligations which apply to DASH-IF documents, as such rights and obligations are set forth and defined in the DASH-IF Bylaws and IPR Policy including, but not limited to, patent and other intellectual property license rights and obligations. A copy of the DASH-IF Bylaws and IPR Policy can be obtained at <http://dashif.org/>.

The material contained herein is provided on an "AS IS" basis and to the maximum extent permitted by applicable law, this material is provided AS IS, and the authors and developers of this material and DASH-IF hereby disclaim all other warranties and conditions, either express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of accuracy or completeness of responses, of workmanlike effort, and of lack of negligence.

In addition, this document may include references to documents and/or technologies controlled by third parties. Those third party documents and technologies may be subject to third party rules and licensing terms. No intellectual

property license, either implied or express, 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.

Note that technologies included in this document and for which no test and conformance material is provided, are only published as a candidate technologies, and may be removed if no test material is provided before releasing a new version of this guidelines document. For the availability of test material, please check <http://www.dashif.org>.

4. DASH and related standards

DASH is a set of manifest and media formats for adaptive media delivery defined by [\[MPEGDASH\]](#). 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. In May 2014, ISO/IEC published the second version of ISO/IEC 23009-1 that includes additional features and provide additional clarifications. ISO/IEC published the third and fourth editions of ISO/IEC 23009-1 in 2019 and 2020.

ISO/IEC also published the 1st and 2nd edition of ISO/IEC 23000-19 'Common media application format (CMAF) for segmented media' [\[MPEGCMAF\]](#) in 2018 and 2019. CMAF defines segment and chunk format based on ISO Base Media File Format, optimized for streaming delivery. CMAF defines a set of well defined constraints that allows interoperability for media deliverable objects, which are compatible with [\[MPEGDASH\]](#).

This document is based on the 4th edition DASH [\[MPEGDASH\]](#) and 2nd edition CMAF [\[MPEGCMAF\]](#) specifications.

DASH together with related standards and specifications is the foundation for an ecosystem of services and clients that work together to enable audio/video/text and related content to be presented to end-users.

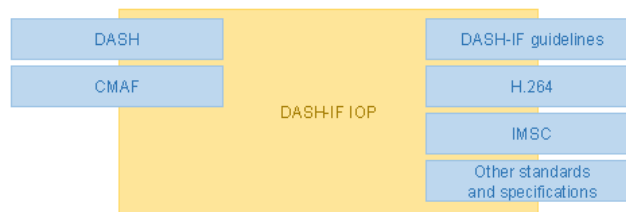


Figure 1 This document connects DASH with international standards, industry specifications and DASH-IF guidelines.

[\[MPEGDASH\]](#) defines a highly flexible set of building blocks that needs to be constrained to a meaningful subset to ensure interoperable behavior in common scenarios. This document defines constraints that limit DASH features to those that are considered appropriate for use in interoperable clients and services.

This document was generated in close coordination with [\[DVB-DASH\]](#). The features are aligned to the extent considered reasonable. The tools and features are aligned to the extent considered reasonable. In addition, DASH-IF worked closely with ATSC to develop a DASH profile for ATSC3.0 for broadcast distribution [\[ATSC3\]](#).

Clients consuming DASH content will need to interact with the host device's media platform. While few constraints are defined on these interactions, this document does assume that the media platform implements APIs that are equivalent to the popular [Media Source Extensions \(MSE\)](#) and [Encrypted Media Extensions \(EME\)](#).

4.1. Relationship to the previous versions of this document

There is no strict backward compatibility with previous versions of this document - best practices change over time and what was once considered sensible may be replaced by a superior approach later on. Therefore, clients and services that were conforming to version N of this document are not guaranteed to conform to version N+1.

The initial two versions of this document were based on the first edition of ISO/IEC 23009-1. Version 4.3 was mostly relying on the third edition of ISO/IEC 23009-1.

This version of the document relies on the 4th edition of ISO/IEC 23009-1 that was technically frozen in July 2019 and is expected to be published by the end of 2019 as ISO/IEC 23009-1:2020.

4.2. Structure of a DASH presentation

[\[MPEGDASH\]](#) specifies the structure of a DASH presentation, which consists primarily of:

1. The manifest or **MPD**, which describes the content and how it can be accessed.

2. Data containers that clients will download over the course of a presentation in order to obtain media samples.

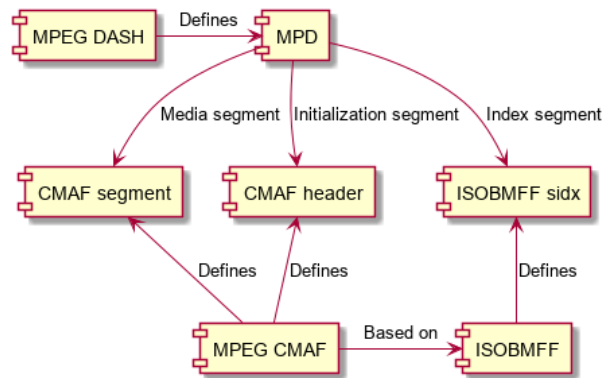


Figure 2 Relationships of primary DASH data structure and the standards they are defined in.

The MPD is an XML file that follows a schema defined by [MPEGDASH]. This schema defines various extension mechanisms for 3rd parties. This document defines some extensions, as do other industry specifications.

[MPEGDASH] defines two data container formats, one based on [ISOBMFF] and the other [MPEG2TS]. However, only the former is used in modern solutions. This document only supports services using the [ISOBMFF] container format.

[MPEGCMAF] is the constrained media format based on [ISOBMFF], specifically designed for adaptive streaming. This document uses [MPEGCMAF] compatible data containers.

Note: The relationship to [MPEGCMAF] is constrained to the container format. In particular, there is no requirement to conform to [MPEGCMAF] media profiles.

The data container format defines the physical structure of the following elements described by the MPD:

1. Each [representation](#) in the [MPD](#) references an [initialization segment](#).
2. Each [representation](#) in the [MPD](#) references any number of [media segments](#).
3. Some [representations](#) in the [MPD](#) may reference an [index segment](#), depending on the [addressing mode](#) used.

Note: HLS (RFC8216) also support ([MPEGCMAF]). Therefore, under certain constraints, the content encoded in ([MPEGCMAF]) can be delivered using MPD or HLS m3u8 manifest format.

[MPEGDASH]	[MPEGCMAF]	[ISOBMFF]
(media) segment, subsegment	CMAF segment	
initialization segment	CMAF header	
index segment, segment index		segment index box (sidx)

Figure 3 Quick reference of closely related terms in different standards.

Note: [MPEGDASH] has the concept of "segment" (URL-addressable media object) and "subsegment" (byte range of URL-addressable media object), whereas [MPEGCMAF] does not make such a distinction. This document uses [MPEGCMAF] segment terminology, with the term segment in this document being equivalent to "CMAF segment" which in turns means "DASH media segment or media subsegment", depending the employed DASH profile.

5. Interoperability requirements

The DASH-related standards enable various options for each feature supported by these standards. Limiting options and in some cases additional constraints are needed to establish interoperable behavior between service offerings and client implementations.

This chapter defines the requirements that enable DASH services and clients to provide interoperable behavior. To

be compliant to a feature in this document, each service offering or client must conform to specific requirements of that feature, outline in this document.

ISSUE 1 Need to add a paragraph on interoperability on baseline, if we have any

5.1. CMAF and ISO BMFF Requirements

Media segments SHALL be compliant to [\[MPEGDASHCMAFPROFILE\]](#).

Note: [\[MPEGDASHCMAFPROFILE\]](#) defines the media segment format using [\[MPEGCMAF\]](#), which is largely based on [\[ISOBMFF\]](#).

5.2. Timing models

The purpose of this chapter is to give a holistic overview of DASH presentation timing and related segment addressing. It is not intended to provide details of the timing model and all possible uses of the attributes in [\[MPEGDASH\]](#).

In order to achieve higher interoperability, DASH-IF's Implementation Guidelines allow considerably limited options than the ones provided by [\[MPEGDASH\]](#), constraining services to a specific set of reasonably flexible behaviors that are highly interoperable with modern client platforms. This chapter covers the timing model and related segment addressing schemes for these common use-cases.

5.2.1. Conformance requirements

This document adds additional constraints to [\[MPEGDASH\]](#) timing requirements.

To be conformant to this document:

- Content generated by a service offering SHALL be compliant to
 - [\[MPEGDASH\]](#) and [\[MPEGDASHCMAFPROFILE\]](#).
 - Additional constraints in following sections
- Clients SHALL be compliant to the constraints in the following sections.

5.2.2. MPD Timeline

[\[MPEGDASH\]](#) defines DASH general timing model in its clause 4.3.

The [MPD](#) defines the **MPD timeline** of a **Media Presentation**, which serves as the baseline for all scheduling decisions made during DASH presentation playback.

There exist two types of Media Presentations, indicated by the `MPD@type`.

The playback of a **static MPD** (defined in [\[MPEGDASH\]](#) as a MPD with `MPD@type="static"`) does not depend on the mapping of the MPD timeline to real time. This means that entire presentation is available at any time and a client can play any part of the presentation at any time (e.g. it can start playback at any time and seek freely within the entire presentation).

The [MPD timeline](#) of a **dynamic MPD** (defined in [\[MPEGDASH\]](#) as a MPD with `MPD@type="dynamic"`) has a fixed mapping to wall clock time, with each point on the [MPD timeline](#) corresponding to a point in real time. This means that segments of the presentation get available over time. Clients can introduce an additional offset with respect to wall clock time for the purpose of maintaining an input buffer to cop with network bandwidth fluctuations.

Note: In addition to mapping the [MPD timeline](#) to wall clock time, [a dynamic MPD can be updated during the presentation](#). Updates may add new [periods](#) and remove or modify existing ones including adding new segments with progress in time, though some restrictions apply. See [§ 5.2.9.5 MPD updates](#).

The time zero on the [MPD timeline](#) of a [dynamic MPD](#) is mapped to the point in wall clock time indicated by `MPD@availabilityStartTime`.

The ultimate purpose of the [MPD](#) is to enable the client to obtain media samples for playback. Additionally it may possibly dynamically switch between different bitrate of the same content to adopt to the network bandwidth fluctuation. The following data structures are most relevant to locating and scheduling the samples:

1. The [MPD](#) consists of consecutive [periods](#) which map data onto the [MPD timeline](#).
2. Each [period](#) contains of one or more [representations](#), each of which provides media samples inside a sequence of [media segments](#).
3. [Representations](#) within a [period](#) are grouped in [adaptation sets](#), which associate related [representations](#) and decorate them with metadata.

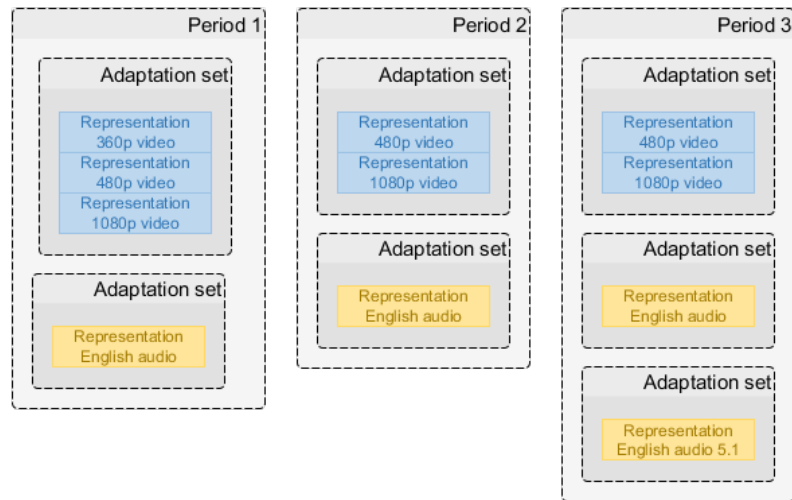


Figure 4 The primary elements described by an [MPD](#).

5.2.3. Periods

An [MPD](#) defines an ordered list of one or more consecutive [periods](#). A [period](#) is both a time span on the [MPD timeline](#) and a definition of the data to be presented during this time span. [Period](#) timing is relative to the zero point of the [MPD timeline](#).



Figure 5 An [MPD](#) is a collection of consecutive periods.

Common reasons for defining multiple [periods](#) are:

- Assembling a presentation from multiple self-contained pieces of content.
- Inserting ads in the middle of existing content and/or replacing spans of existing content with ads.
- Adding/removing certain [representations](#) as the nature of the content changes (e.g. a new title starts with a different set of offered languages).
- Updating period-scoped metadata (e.g. codec configuration or DRM signaling).

[Periods](#) are self-contained - a client is not required to know the contents of another [period](#) in order to correctly present a [period](#). Knowledge of the contents of different periods may be used by a client to achieve seamless [period](#) transitions, especially when working with [period-connected representations](#).

EXAMPLE 1

The below [static MPD](#) consists of two 20-second [periods](#). The duration of the first [period](#) is calculated using the start point of the second [period](#). The total duration of the presentation is 40 seconds.

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" type="static">
  <Period>
    ...
  </Period>
  <Period start="PT20S" duration="PT20S">
    ...
  </Period>
</MPD>
```

Parts of the [MPD](#) structure that are not relevant for this chapter have been omitted - this is not a fully functional [MPD](#) file.

[\[MPEGDASH\]](#) clause 5.3.2 defines the period's requirements in MPD authoring. Among others it requires the followings:

1. All periods are consecutive and non-overlapping. A [period](#) may have a duration of zero.

Note: A [period](#) with a duration of zero might, for example, be the result of ad-insertion logic deciding not to insert any ad.

2. In a [static MPD](#), the first [period](#) starts at the time zero of the [MPD timeline](#). In a [dynamic MPD](#), the first [period](#) starts at or after the zero point of the [MPD timeline](#).
3. In a [static MPD](#), either the last [period](#) has a `Period@duration` or `MPD@mediaPresentationDuration` exists.
4. In a [dynamic MPD](#), the last [period](#) may have a `Period@duration`, in which case it has a fixed duration. If without `Period@duration`, the last [period](#) in a [dynamic MPD](#) has an unknown duration, which allows to extend the timeline indefinitely.

Note: In a [dynamic MPD](#), a [period](#) with an unknown duration may be converted to fixed-duration by an MPD update. Periods in a [dynamic MPD](#) can also be shortened or removed entirely under certain conditions. However, [Media Presentation](#) is defined until (current wall clock time + `MPD@minimumUpdatePeriod`), by which the current MPD is still valid. See [§ 5.2.9.5 MPD updates](#).

5. `MPD@mediaPresentationDuration` may be present. If present, it accurately matches the duration between the time zero on the [MPD timeline](#) and the end of the last period. Clients must calculate the total duration of a [static MPD](#) by adding up the durations of each [period](#) and must rely on the presence of `MPD@mediaPresentationDuration`.

Note: This calculation is necessary because the durations of [XLink periods](#) can only be known after the [XLink](#) is resolved. Therefore it is impossible to always determine the total [MPD](#) duration on the service side as only the client is guaranteed to have access to all the required knowledge.

5.2.4. Representations

A **representation** is a sequence of **segment** as defined by [\[MPEGDASH\]](#) 5.3.1 and 5.3.5. A `Representation` element is a collection of these **segment references** and a description of the samples within the referenced [media segments](#).

In practice, each representation usually belongs to exactly one [adaptation set](#) and often belongs to exactly one [period](#), although a [representation may be connected with a representation in another period](#).

Each [segment](#) reference addresses a [media segment](#) that corresponds to a specific time span on the [sample timeline](#). Each [media segment](#) contains samples for a specific time span on the [sample timeline](#).

Note: [Simple addressing](#) allows the actual time span of samples within a [media segment](#) to deviate from the corresponding time span described in the [MPD](#) ([\[MPEGDASH\]](#) 7.2.1). All timing-related clauses in this document refer to the timing described in the [MPD](#) (i.e. according to [MPD timeline](#)) unless otherwise noted.

The exact mechanism used to define segment references depends on the [addressing mode](#) used by the representation.

This document requires the following additional requirement:

- All [representations](#) in the same [adaptation set](#) SHALL use the same [addressing mode](#).

As recommended by [\[MPEGDASH\] 7.2.1](#):

- There should not be gaps or overlapping [media segments](#) in a [representation](#).

This document additionally requires:

- In a [static MPD](#) a [representation](#) SHALL contain enough [segment references](#) to cover the entire time span of the [period](#).

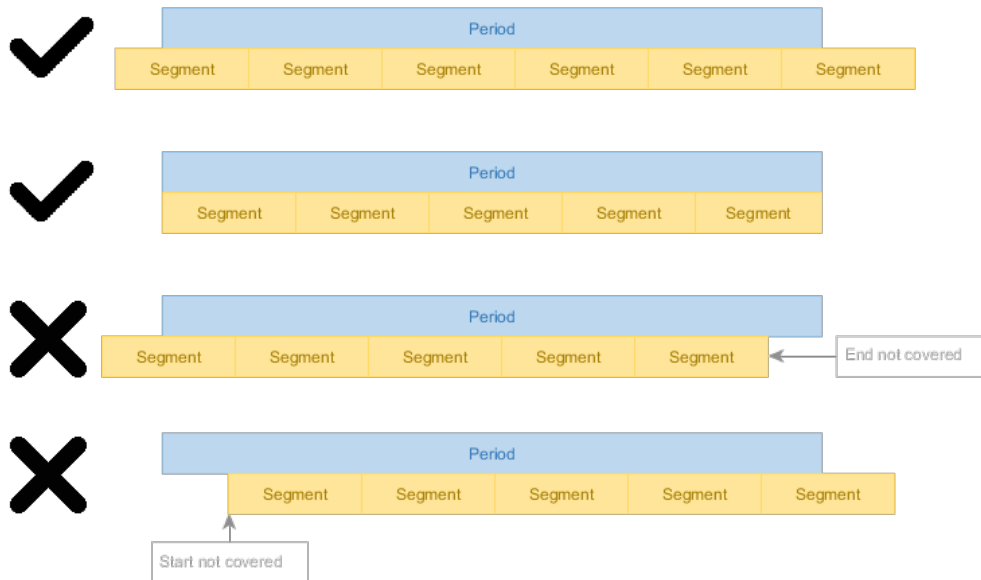


Figure 6 In a [static MPD](#), the entire [period](#) must be covered with [media segments](#).

- In a [dynamic MPD](#), a [representation](#) element SHALL contain enough [segment references](#) to cover the time span of the [period](#) that intersects with the [time shift buffer](#). However, gaps in this time span are allowed.

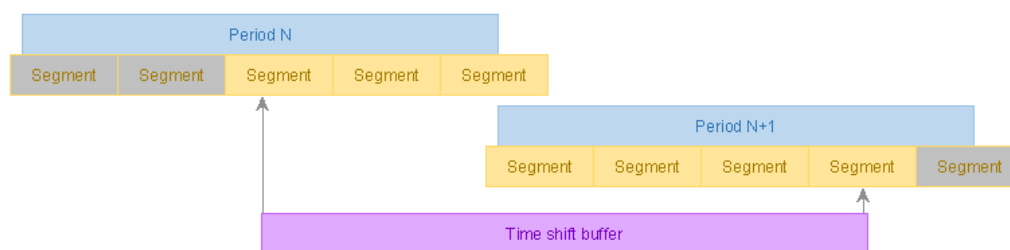


Figure 7 In a [dynamic MPD](#), the [time shift buffer](#) determines the set of required [segment references](#) in each [representation](#). [Media segments](#) filled with gray need not be referenced due to falling outside the [time shift buffer](#), despite falling within the bounds of a [period](#).

Note: In a dynamic MPD, each [Media segments](#) only become [available](#) when its end point is within their [availability window](#) (This time may need to be adjusted by `availabilityTimeOffset` (need to be defined) and `@availabilityTimeComplete` values) ([\[MPEGDASH\] 5.3.9.5.1](#) and [5.3.5.3](#)). It is a valid situation that a [media segment](#) is required to be referenced but is not yet [available](#).

As required by [\[MPEGDASH\] 5.3.9.5.3](#):

- A [dynamic MPD](#) remains valid for its entire validity duration after publishing. In other words, a [dynamic MPD](#) supplies enough [segment references](#) to allow the [time shift buffer](#) to extend to `now + MPD@minimumUpdatePeriod`, where `now` is the current time according to [the synchronized clock](#).

As allowed by [\[MPEGDASH\] 7.2.1](#):

- [Media segment](#) start/end points may be unaligned with [period](#) start/end points except when using [simple addressing](#). This possible offset is signaled by `@eptDelta`.

An **unnecessary segment reference** is one that is not defined as required by this chapter.

This document requires the following additional requirements to [\[MPEGDASH\]](#):

- In a [static MPD](#), a [representation](#) SHALL NOT contain [unnecessary segment references](#), except when using [indexed addressing](#) in which case such segment references MAY be present.
- In a [dynamic MPD](#), a [representation](#) SHALL NOT contain [unnecessary segment references](#) except when any of the following applies, in which case an [unnecessary segment reference](#) MAY be present:
 1. The [segment reference](#) is for future content and will eventually become necessary.
 2. The [segment reference](#) is defined via [indexed addressing](#).
 3. The [segment reference](#) is defined by an <s> element that defines multiple references using s@r, some of which are necessary.
 4. Removal of the [segment reference](#) is not allowed by [content removal constraints](#).

This document also requires the following requirements for clients:

- Clients SHALL NOT present any samples from [media segments](#) that are entirely outside the [period](#), even if such [media segments](#) are referenced.

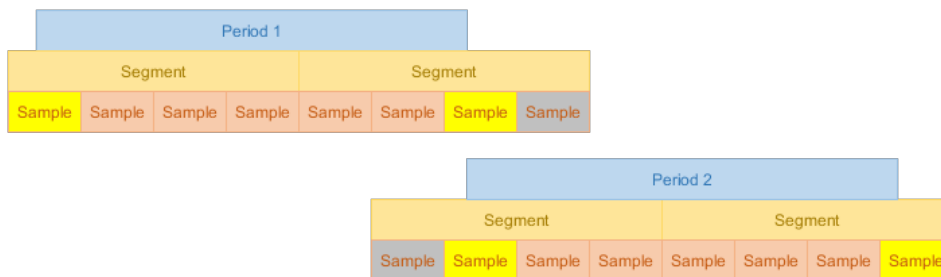


Figure 8 [Media segments](#) and samples need not align with [period](#) boundaries. Some samples may be entirely outside a [period](#) (marked gray) and some may overlap the [period](#) boundary (yellow).

- If a [media segment](#) overlaps a [period](#) boundary, clients SHOULD NOT present the samples that lie outside the [period](#) and SHOULD present the samples that lie either partially or entirely within the [period](#).

Note: In the end, which samples are presented is entirely up to the client. It may sometimes be impractical to present [media segments](#) only partially, depending on the capabilities of the client platform, the type of media samples involved and any dependencies between samples.

5.2.5. Sample timeline

The samples within a [representation](#) exist on a linear **sample timeline** defined by the encoder that created the samples. One or more [sample timelines](#) are mapped onto the [MPD timeline](#) by metadata stored in or referenced by the [MPD](#) ([\[MPEGDASH\]](#) 7.3.2).

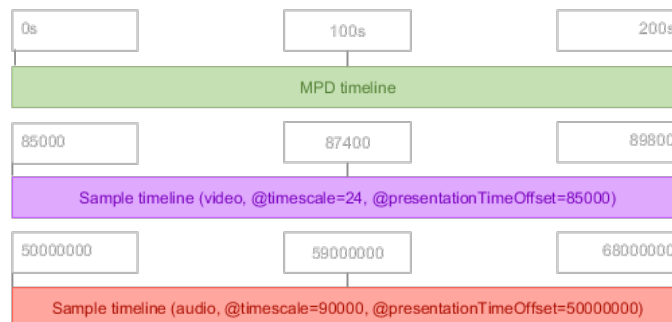


Figure 9 [Sample timelines](#) are mapped onto the [MPD timeline](#) based on parameters defined in the [MPD](#).

Note: A [sample timeline](#) is linear - encoders are expected to use an appropriate [timescale](#) and sufficiently large timestamp fields to avoid any wrap-around. If wrap-around does occur, a new [period](#) must be started in order to establish a new [sample timeline](#).

The [sample timeline](#) is formed after applying any [\[ISOBMFF\]](#) edit lists ([\[MPEGDASH\]](#) 7.3.2).

This document additionally requires:

- The same [sample timeline](#) SHALL be shared by all [representations](#) in the same [adaptation set](#). [Representations](#) in different [adaptation sets](#) MAY use different [sample timelines](#).
- The [sample timeline](#) is measured in **timescale units** defined as a number of units per second. This value (the **timescale**) SHALL be present in the MPD as `SegmentTemplate@timescale` or `SegmentBase@timescale` (depending on the [addressing mode](#)).

Note: While optional in [\[MPEGDASH\]](#), the presence of the `@timescale` attribute is required by the interoperable timing model because the default value of 1 is unlikely to match any real-world content and is far more likely to indicate an unintentional content authoring error.

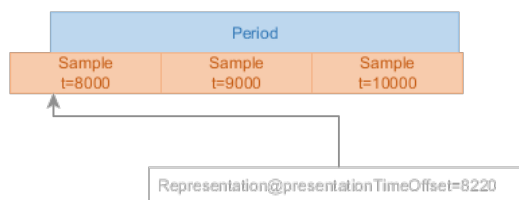


Figure 10 `@presentationTimeOffset` is the key component in establishing the relationship between the [MPD timeline](#) and a [sample timeline](#).

The point on the [sample timeline](#) indicated by `@presentationTimeOffset` is equivalent to the [period](#) start point on the [MPD timeline](#) ([\[MPEGDASH\]](#) Table 15). The value is provided by `SegmentTemplate@presentationTimeOffset` or `SegmentBase@presentationTimeOffset`, depending on the [addressing mode](#), and has a default value of 0 [timescale units](#).

Note: To transform a [sample timeline](#) position `SampleTime` to an [MPD timeline](#) position, use the formula $MpdTime = Period@start + (SampleTime - @presentationTimeOffset) / @timescale$.

5.2.6. Clock drift is forbidden

Some encoders experience clock drift - they do not produce exactly 1 second worth of output per 1 second of input, either stretching or compressing the [sample timeline](#) with respect to the [MPD timeline](#).

This document adds the following requirement:

- A DASH service SHALL NOT publish content that suffers from clock drift.

If a packager receives input from an encoder at the wrong rate, it must take corrective action. For example, it might:

1. Drop a span of content if input is produced faster than real-time.
2. Insert regular padding content if input is produced slower than real-time. This padding can take different forms:
 - Silence or a blank picture.
 - Repeating frames.
 - Insertion of short-duration [periods](#) where the affected [representations](#) are not present.

Of course, such after-the-fact corrective actions can disrupt the end-user experience. The optimal solution is to fix the defective encoder.

5.2.7. Media segments

A **media segment** is an HTTP-addressable data structure that contains one or more media samples.

Note: Different media segments may be different byte ranges accessed on the same URL.

[\[MPEGCMF\]](#) requires that [Media segments](#) contain one or more consecutive media samples, and consecutive [media segments](#) in the same [representation](#) contain consecutive media samples.

[\[MPEGDASH\]](#) 7.2.1 requires the followings:

- [Media segments](#) contains the media samples that exactly match the time span on the [sample timeline](#) that is assigned to the [media segment](#) by the MPD, except when using [simple addressing](#) in which case a certain amount of inaccuracy may be present as defined in § 5.3.4.1 [Inaccuracy in media segment timing when using simple addressing](#).
- The [media segment](#) that starts at or overlaps the [period](#) start point contains a media sample that starts at or overlaps the [period](#) start point and the [media segment](#) that ends at or overlaps the [period](#) end point contains a media sample that ends at or overlaps the [period](#) end point.

[MPEGCMAF] 7.3.4 and [MPEGDASHCMAFPROFILE] requires the following:

- [Aligned media segments](#) in different [representations](#) of the same adaptation set contains samples for the same true time span, even if using [simple addressing](#) with [inaccurate media segment timing](#).

5.2.7.1. Media segment duration deviation§

When using [simple addressing](#), the samples contained in a media segment may cover a different time span on the [sample timeline](#) than what is indicated by the nominal timing in the [MPD timeline](#). This deviation is defined as the offset between the edges of the nominal time span (as defined by [MPD timeline](#)) and the edges of the true time span (as defined by [sample timeline](#)), and is calculated separately for each edge.

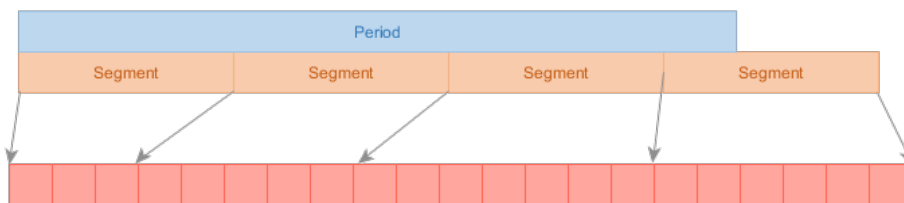


Figure 11 In simple addressing, a media segment may cover a different time span on the [sample timeline](#) than what is indicated by the nominal timing in the [MPD timeline](#). Red boxes indicate samples.

[MPEGDASH] 7.2.1 requires: The duration deviation is no more than 50% of the nominal media segment duration and may be in either direction.

This document also recommends:

- [Media segments](#) of a [representation](#) SHOULD be equal in duration. Occasional jitter MAY occur (e.g. due to encoder decisions on GOP size).

Note: [DVB-DASH] defines some relevant constraints in section 4.5. Consider obeying these constraints to be compatible with [DVB-DASH].

5.2.7.2. Segments must be aligned§

[Media segments](#) are said to be aligned if the earliest presentation time of all [media segments](#) on the [sample timeline](#) is equal in all [representations](#) that belong to the same [adaptation set](#).

[MPEGDASHCMAFPROFILE] requires:

- [Media segments](#) are aligned.
- When using [simple addressing](#) or [explicit addressing](#), the media segments alignment is signaled by `AdaptationSet@segmentAlignment=true` in the [MPD](#). When using [indexed addressing](#), this is signaled by `AdaptationSet@subsegmentAlignment=true` in the [MPD](#).

5.2.8. Period connectivity§

The precise definition of Period connectivity can be found in [MPEGDASH] 5.3.2.4. However, generally speaking, in certain circumstances content may be offered such that a [representation](#) is technically compatible with the content of a [representation](#) in a previous [period](#). Such [representations](#) are **period-connected**.

Any subset of the [representations](#) in a [period](#) may be [period-connected](#) with their counterparts in a future or past [period](#). [Period](#) connectivity may be chained across any number of [periods](#).

Note: Connectivity is generally achieved by using the same encoder to encode the content of multiple [periods](#) using the same settings. Keep in mind, however, that decryption is also a part of the client media pipeline - it is not only the codec parameters that are configured by the initialization segment; different decryption parameters are likely to break connectivity that would otherwise exist.

For signaling the period connectivity between [representation](#) of two periods in a MPD, [\[MPEGDASH\] 5.3.2.4](#) requires:

- `Representation@id` is equal.
- `AdaptationSet@id` is equal.
- The [adaptation set](#) in the second [period](#) has a [supplemental property descriptor](#) with:
 - `@schemeIdUri` set to `urn:mpeg:dash:period-connectivity:2015`.
 - `@value` set to the `Period@id` of the first period.
- Initialization segments of period-connected [representations](#) to be functionally equivalent (i.e. the initialization segment from any period-connected [representation](#) can be used to initialize playback of any period-connected [representation](#)).

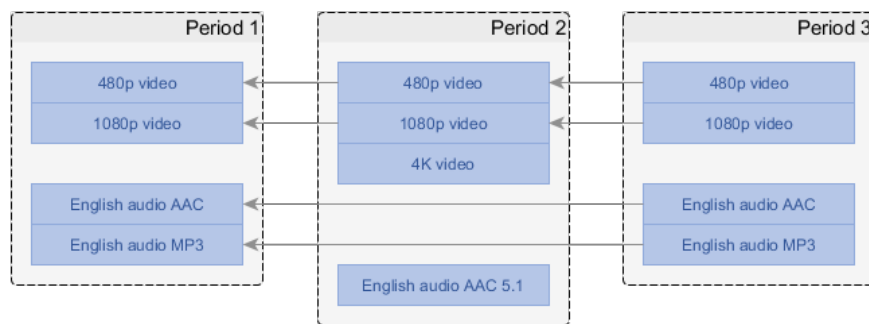


Figure 12 [Representations](#) can be signaled as [period-connected](#), enabling client optimizations. Arrows on diagram indicate direction of connectivity reference (from future to past), with the implied message being "the client can use the same decoder it used where the arrow points to".

Note: Not all [representations](#) in an [adaptation set](#) need to be [period-connected](#). For example, if a new [period](#) is introduced to add a [representation](#) that contains a new video quality level, all other [representations](#) will likely be connected but not the one that was added.

Note that [\[MPEGDASH\]](#) allows:

- An [MPD](#) may contain unrelated [periods](#) between [periods](#) that contain [period-connected representations](#).
- The [sample timelines](#) of [period-connected representations](#) may be mutually discontinuous (e.g. due to encoder clock wrap-around or skipping some content as a result of editorial decisions).
- As a [period](#) may start and/or end in the middle of a [media segment](#), the same [media segment](#) may simultaneously exist in two [period-connected representations](#), with one part of it scheduled for playback during the first [period](#) and the other part during the second [period](#). This is likely to be the case when no [sample timeline](#) discontinuity is introduced by the transition.

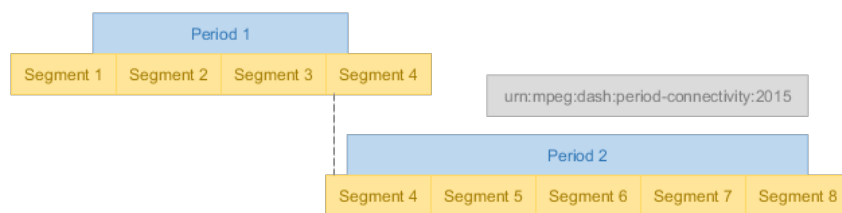


Figure 13 The same [media segment](#) will often exist in two [periods](#) at a [period-connected](#) transition. On the diagram, this is [segment 4](#).

This document recommends:

- [Media Presentation](#) with connected content cross periods SHOULD be signaled in the [MPD](#) as [period-connected](#). This is expected to help clients ensure seamless playback across [period](#) transitions.

This document also recommends:

- Clients SHOULD NOT present a [media segment](#) twice when it occurs on both sides of a [period](#) transition in a [period-connected representation](#).
- Clients SHOULD ensure seamless playback of [period-connected representations](#) in consecutive [periods](#).

Note: The exact mechanism that ensures seamless playback depends on client capabilities and will be implementation-specific. Any shared [media segment](#) overlapping the [period](#) boundary may need to be detected and deduplicated to avoid presenting it twice.

5.2.8.1. Period continuity⁵

In addition to [period connectivity](#), [MPEGDASH] 5.3.2.4 defines [period](#) continuity, which is a special case of [period](#) connectivity where the two samples on the boundary between the connected [representations](#) are consecutive on the same [sample timeline](#). Continuity implies connectivity.

Note: The above can only be true if the sample boundary exactly matches the [period](#) boundary.

For signaling the period continuity, [MPEGDASH] 5.3.2.4 requires:

- The same signaling as for [period connectivity](#), except that the value to use for @schemeIdUri is urn:mpeg:dash:period-continuity:2015.

This document requires:

- [Media Presentation](#) with continuous content cross periods SHOULD be signaled in the [MPD](#) with period continuity.
- [period](#) connectivity SHALL NOT be simultaneously signaled on the same [representation](#) for which period continuity is signaled.

This document requires:

- Clients MAY take advantage of any platform-specific optimizations for seamless playback that knowledge of [period](#) continuity enables; beyond that, clients SHALL treat continuity the same as connectivity.

5.2.9. Dynamic MPDs⁵

This section only applies to [dynamic MPDs](#).

Three main factors differentiate them from [static MPDs](#):

1. The segments described in a [dynamic MPD](#) may become available over time, i.e. not all segments are available.
2. Playback of a [dynamic MPD](#) is synchronized to a real time clock (with some amount of client-chosen time shift allowed).
3. A [dynamic MPD](#) may change over time, with clients retrieving new snapshots of the [MPD](#) when the validity duration of the previous snapshot expires.

[MPEGDASH] 5.4.1 requires:

- A dynamic MPD conforms to the MPD constraints not only at its moment of initial publishing but through the entire **MPD validity duration**, which is a period of MPD@minimumUpdatePeriod starting from the moment the MPD download is started by a client, unless overridden by [in-band validity expiration signaling](#).

The [MPD validity duration](#) starts when the MPD download is initiated by a client, which may be some time after it is generated/published!

This document requires: DASH clients SHALL support the presentation of [dynamic MPDs](#).

5.2.9.1. Real time clock synchronization⁵

It is critical to synchronize the clocks of the client with the clock of service when using a [dynamic MPD](#). The time indicated by the clock does not necessarily need to match some universal standard as long as the two are mutually synchronized.

The use of UTCTiming is optional in [MPEGDASH].

This document requires:

- A [dynamic MPD](#) SHALL include at least one `UTCTiming` element that defines a clock synchronization mechanism. If multiple `UTCTiming` elements are listed, their order determines the order of preference.
- The set of time synchronization mechanisms SHALL be restricted to the following schemes defined in [MPEGDASH]:
 - `urn:mpeg:dash:utc:http-xsdate:2014`
 - `urn:mpeg:dash:utc:http-iso:2014`
 - `urn:mpeg:dash:utc:http-ntp:2014`
 - `urn:mpeg:dash:utc:ntp:2014`
 - `urn:mpeg:dash:utc:http-head:2014`
 - `urn:mpeg:dash:utc:direct:2014`

The use of a "default time source" is not allowed. The mechanism of time synchronization must always be explicitly defined in the MPD by every service.

This document requires:

- A client presenting a [dynamic MPD](#) SHALL synchronize its local clock according to the `UTCTiming` elements in the [MPD](#) and SHALL emit a warning or error to application developers when clock synchronization fails, no `UTCTiming` elements are defined or none of the referenced clock synchronization mechanisms are supported by the client.

ISSUE 2 We could benefit from some detailed examples here, especially as clock sync is such a critical element of live services.

5.2.9.2. Availability

A [media segment](#) is **available** when an HTTP request to acquire the [media segment](#) can be started and successfully performed to completion by a client. During playback of a [dynamic MPD](#), new [media segments](#) continuously become [available](#) and stop being [available](#) with the passage of time. [MPEGDASH] defines the **segment availability times** of a segment as the duration in wall-clock time in which that segment is available.

An **availability window** is a time span on the [MPD timeline](#) that determines which [media segments](#) can be expected to be [available](#). Each [representation](#) has its own [availability window](#). Consequently, [availability window](#) at each moment is defined by the union of [segment availability times](#) of all available segments at that moment.

A **segment start point** (referred to as **MPD start time** of a segment in [MPEGDASH]) is the presentation start time of the segment in [MPD timeline](#).

A **segment end point** is defined as the presentation end time of the segment in [MPD timeline](#).

[MPEGDASH] requires:

- A service makes [available](#) all [media segments](#) that have their end point inside or at the end of the [availability window](#).

It is the responsibility of the service to ensure that [media segments](#) are [available](#) to clients when they are described as [available](#) by the [MPD](#). Consider that the criterium for availability is a successful download by clients, not successful publishing from a packager.

The [availability window](#) is calculated as follows:

1. Let *now* be the current wall clock time according to [the synchronized clock](#).
2. Let *AvailabilityWindowStart* be $now - MPD@timeShiftBufferDepth$.
 - If `MPD@timeShiftBufferDepth` is not defined, let *AvailabilityWindowStart* be `MPD@availabilityStartTime`.
3. Let *TotalAvailabilityTimeOffset* be the sum of all `@availabilityTimeOffset` values that apply to the [representation](#) (those directly on the `Representation` element and any of its ancestors).
4. The [availability window](#) is the time span from *AvailabilityWindowStart* to $now + TotalAvailabilityTimeOffset$.

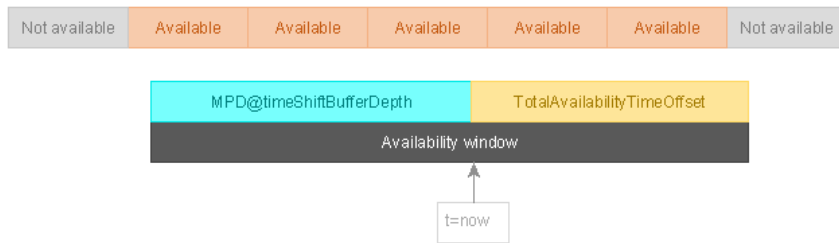


Figure 14 The availability window determines which media segments can be expected to be available, based on where their segment end point lies.

This document requires:

- Clients MAY at any point attempt to acquire any media segments that the MPD signals as available. Clients SHALL NOT attempt to acquire media segments that the MPD does not signal as available.
- Clients SHOULD NOT assume that media segments described by the MPD as available are available and SHOULD implement appropriate retry/fallback behavior to account for timing errors by slow-publishing or eagerly-unpublishing services.

5.2.9.3. Time shift buffer

The **time shift buffer** is a time span on the MPD timeline that defines the set of media segments that a client is allowed to present at the current moment in time according to the synchronized clock (now).

This is the mechanism by which clients can introduce a **time shift** (an offset) between real time and the MPD timeline when presenting dynamic MPDs. The time shift is zero when a client always chooses to play back the media segment at the end point of the time shift buffer. By playing back media segments from further in the past, a time shift is introduced.

Note: A time shift of 30 seconds means that the client starts presenting a media segment at the moment when its position on the MPD timeline reaches a distance of 30 seconds from the end of the time shift buffer.

The following additional factors further constrain the set of media segments that can be presented at the current time and can force a client to introduce a time shift:

1. § 5.2.9.2 Availability - not every media segment in the time shift buffer is guaranteed to be available.
2. § 5.2.9.4 Presentation delay - the service may define a delay that forbids the use of a section of the time shift buffer.

The time shift buffer extends from `now - MPD@timeShiftBufferDepth` to `now`. In the absence of `MPD@timeShiftBufferDepth` the start of the time shift buffer is `MPD@availabilityStartTime`.

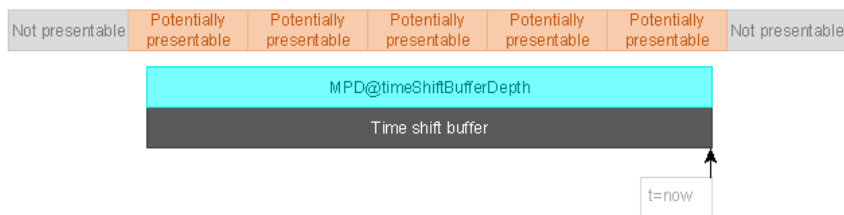


Figure 15 Media segments overlapping the time shift buffer may potentially be presented by a client, if other constraints do not forbid it.

This document requires:

- Clients MAY present samples from media segments that overlap (either in full or in part) the time shift buffer, assuming no other constraints forbid it.
- Clients SHALL NOT present samples from media segments that are entirely outside the time shift buffer (whether in the past or the future).
- The start of the time shift buffer may be before the start of the first period. Clients SHALL NOT use regions of the time shift buffer that are not covered by periods.

A dynamic MPD SHALL contain a period that ends at or overlaps the end point of the time shift buffer, except when reaching the end of live content in which case the last period MAY end before the end of the time shift buffer.

5.2.9.4. Presentation delay

There is a natural conflict between the [availability window](#) and the [time shift buffer](#). It is legal for a client to present [media segments](#) as soon as they overlap the [time shift buffer](#), yet such [media segments](#) might not yet be [available](#).

Furthermore, the delay between [media segments](#) entering the [time shift buffer](#) and becoming [available](#) might be different for different [representations](#) that use different [media segment](#) durations. This difference may also change over time if a [representation](#) does not use a constant [media segment](#) duration.

This document requires:

- Clients SHALL calculate a suitable **presentation delay** to ensure that the [media segments](#) it schedules for playback are [available](#) and that there is sufficient time to download them once they become [available](#). In essence, the [presentation delay](#) decreases the [time shift buffer](#), creating an [effective time shift buffer](#) with a reduced duration.

[MPEGDASH] allows:

- Services may define the `MPD@suggestedPresentationDelay` attribute to provide a suggested [presentation delay](#).

This document requires:

- Clients SHOULD use `MPD@suggestedPresentationDelay` when provided, ignoring the calculated value.

Note: As different clients might use different algorithms for calculating the presentation delay, providing `MPD@suggestedPresentationDelay` enables services to roughly synchronize the playback start position of clients.

The **effective time shift buffer** is the time span from the start of the [time shift buffer](#) to `now - PresentationDelay`.

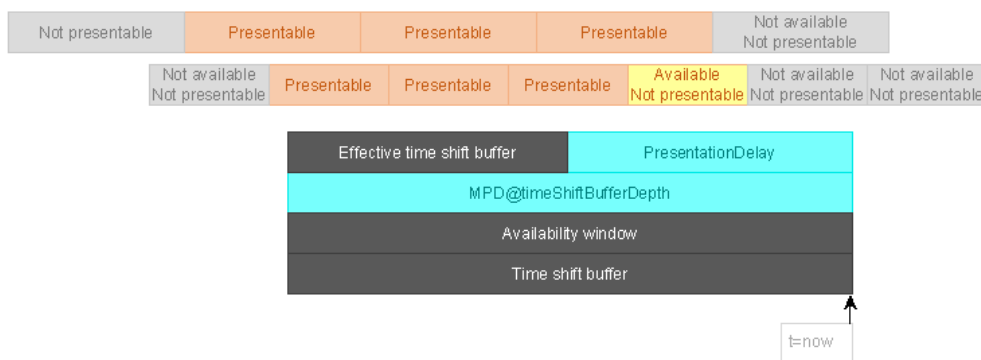


Figure 16 [Media segments](#) that overlap the [effective time shift buffer](#) are the ones that may be presented at time `now`. Two [representations](#) with different segment lengths are shown. Diagram assumes `@availabilityTimeOffset=0`.

This document requires:

- Clients SHALL constrain seeking to the [effective time shift buffer](#). Clients SHALL NOT attempt to present [media segments](#) that fall entirely outside the [effective time shift buffer](#).

5.2.9.5. MPD updates

[Dynamic MPDs](#) may change over time. The nature of the change is not restricted unless such a restriction is explicitly defined.

Some common reasons to make changes in [dynamic MPDs](#):

- Adding new [segment references](#) to an existing [period](#).
- Adding new [periods](#).
- Converting unlimited-duration [periods](#) to fixed-duration [periods](#) by adding `Period@duration`.
- Removing [segment references](#) and/or [periods](#) that have fallen out of the [time shift buffer](#).
- Shortening an existing [period](#) when changes in content scheduling take place.
- Removing `MPD@minimumUpdatePeriod` to signal that [MPD](#) will no longer be updated.

- Converting the [MPD](#) to a [static MPD](#) to signal that a live service has become available on-demand as a recording.

[MPEGDASH] 5.4.1 requires the following restrictions for MPD updates:

- `MPD@id` does not change.
- `MPD.Location` does not change.
- `MPD@availabilityStartTime` does not change.
- `Period@id` does not change.
- `Period@start` does not change.
- `Period@duration` does not change except when explicitly allowed by other statements in this document.
- The [adaptation sets](#) present in a [period](#) (i.e. the set of `AdaptationSet@id` values) does not change.
- The [representations](#) present in an [adaptation set](#) (i.e. the set of `Representation@id` values) does not change.
- The functional behavior of a [representation](#) (identified by a matching `Representation@id` value) does not change, neither in terms of metadata-driven behavior (including metadata inherited from [adaptation set](#) level) nor in terms of [media segment](#) timing. In particular:
 - `SegmentTemplate@presentationTimeOffset` does not change.
 - `SegmentBase@presentationTimeOffset` does not change.

Additional restrictions on MPD updates are defined by other parts of this document.

This document requires:

- The presence or absence of `MPD@minimumUpdatePeriod` SHALL be used by a service to signal whether the MPD might be updated (with presence indicating potential for future updates). The value of this field indicates the [MPD validity duration](#) of the present snapshot of the [MPD](#), starting from the moment its download was initiated. Absence of the `MPD@minimumUpdatePeriod` attribute indicates an infinite validity (the [MPD](#) will never be updated). The value 0 indicates that the [MPD](#) has no validity after the moment it was retrieved.
- Since clients usually require some time to download and process an [MPD](#) update, a service SHOULD NOT assume perfect update timing.
- In addition to signaling that clients are expected to poll for regular [MPD](#) updates, a service MAY publish [in-band events to update the MPD validity duration](#) at moments of its choosing.

This document also requires:

- Clients SHOULD use `@id` to track [period](#), [adaptation set](#) and [representation](#) identity across MPD updates.
- Clients SHALL process state changes that occur during the [MPD validity duration](#). For example new [media segments](#) will become [available](#) over time if they are referenced by the [MPD](#) and old ones become unavailable, even without an [MPD](#) update.
- `MPD@minimumUpdatePeriod = 0` indicates that the [MPD](#) has no validity after the moment it was retrieved. In such a situation, the client SHALL have to acquire a new [MPD](#) whenever it wants to make new [media segments](#) available (no "natural" state changes will occur).
- Clients SHOULD NOT assume that they can get all updates in time (they may already be attempting to buffer some [media segments](#) that were removed by an [MPD](#) update).

5.2.9.5.1. Adding content to the MPD§

[MPEGDASH] allows two mechanisms for adding content:

- Additional [segment references](#) may be added to the last [period](#).
- Additional [periods](#) may be added to the end of the MPD.

Multiple content adding mechanisms may be combined in a single [MPD](#) update. An [MPD](#) update that adds content may be combined [with an MPD update that removes content](#).

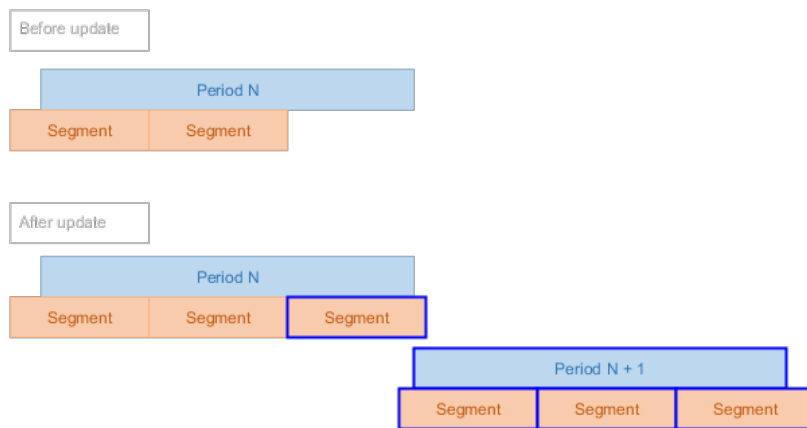


Figure 17 MPD updates can add both [segment references](#) and [periods](#) (additions highlighted in blue).

This document requires:

- [Segment references](#) SHALL NOT be added to any [period](#) other than the last [period](#).
- An MPD update MAY combine adding [segment references](#) to the last [period](#) with adding of new [periods](#).

Note: The duration of the last [period](#) cannot change as a result of adding [segment references](#). A live service will generally use a [period](#) with an unlimited duration to continuously add new [segment references](#).

When using [simple addressing](#) or [explicit addressing](#), it is possible for a [period](#) to define an infinite sequence of [segment references](#) that extends to the end of the [period](#) (e.g. using `SegmentTemplate@duration` or `r="-1"`). Such self-extending reference sequences are equivalent to explicitly defined [segment reference](#) sequences that extend to the end of the [period](#) and clients MAY obtain new [segment references](#) from such sequences even between [MPD](#) updates.

5.2.9.5.2. Removing content from the MPD §

Removal of content is only allowed if the content to be removed is not yet [available](#) to clients and guaranteed not to become [available](#) until clients receive the [MPD](#) update. See § 5.2.9.2 Availability.

To determine the content that may be removed, let `EarliestRemovalPoint` be `availability window end + MPD@minimumUpdatePeriod`.

Note: As each [representation](#) has its own [availability window](#), so does each [representation](#) have its own `EarliestRemovalPoint`.

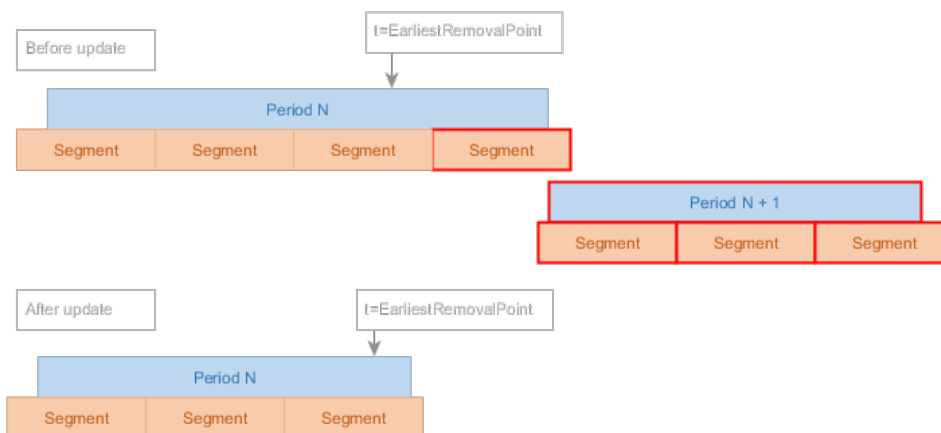


Figure 18 MPD updates can remove both [segment references](#) and [periods](#) (removals highlighted in red).

An [MPD](#) update removing content MAY remove any [segment references](#) to [media segments](#) that start after `EarliestRemovalPoint` at the time the update is published.

[Media segments](#) that overlap or end before `EarliestRemovalPoint` might be considered by clients to be [available](#) at the time the [MPD](#) update is processed and therefore SHALL NOT be removed by an [MPD](#) update.

The following mechanisms exist removing content:

- The last [period](#) MAY change from unlimited duration to fixed duration.
- The duration of the last [period](#) MAY be shortened.
- One or more [periods](#) MAY be removed entirely from the end of the [MPD](#).

Multiple content removal mechanisms MAY be combined in a single [MPD](#) update.

Note: When using [indexed addressing](#) or [simple addressing](#), removal of [segment references](#) from the end of the [period](#) only requires changing `Period@duration`. When using [explicit addressing](#), pruning some `S` elements may be appropriate to avoid leaving [unnecessary segment references](#).

Clients SHALL NOT fail catastrophically if an [MPD](#) update removes already buffered data but MAY incur unexpected [time shift](#) or a visible transition at the point of removal. It is the responsibility of the service to avoid removing data that may already be in use.

In addition to editorial removal from the end of the [MPD](#), content naturally expires due to the passage of time. Expired content also needs to be removed:

- Explicitly defined [segment references](#) (`S` elements) SHALL be removed when they have expired (i.e. the [media segment](#) end point has fallen out of the [time shift buffer](#)).
 - A repeating explicit [segment reference](#) (`S` element with `@r != 0`) SHALL NOT be removed until all repetitions have expired.
- [Periods](#) with their end points before the time shift buffer SHALL be removed.

An [MPD](#) update that removes content MAY be combined [with an MPD update that adds content](#).

5.2.9.5.3. End of live content§

Live services can reach a point where no more content will be produced - existing content will be played back by clients and once they reach the end, playback will cease.

This document requires:

- When this occurs, services SHALL define a fixed duration for the last [period](#), remove the `MPD@minimumUpdatePeriod` attribute and cease performing [MPD](#) updates to signal that no more content will be added to the [MPD](#).
- The `MPD@type` MAY be changed to `static` at this point or later if the service is to be converted to a [static MPD](#) for on-demand viewing.

5.2.9.6. MPD refreshes§

To stay informed of the [MPD](#) updates, clients need to perform **MPD refreshes** at appropriate moments to download the updated [MPD](#) snapshots.

This document requires:

- Clients presenting [dynamic MPDs](#) SHALL execute the following [MPD](#) refresh logic:
 1. When an [MPD](#) snapshot is downloaded, it is valid for the present moment and at least `MPD@minimumUpdatePeriod` after that.
 2. A client can expect to be able to successfully download any [media segments](#) that the [MPD](#) defines as [available](#) at any point during the [MPD validity duration](#).
 3. The clients MAY refresh the [MPD](#) at any point. Typically this will occur because the client wants to obtain more [segment references](#) or make more [media segments](#) (for which it might already have references) [available](#) by extending the [MPD](#) validity duration.
 - This may result in a different [MPD](#) snapshot being downloaded, with updated information.
 - Or it may be that the [MPD](#) has not changed, in which case its validity period is extended to `now + MPD@minimumUpdatePeriod`.

Note: There is no requirement that clients poll for updates at `MPD@minimumUpdatePeriod` interval. They can do so as often or as rarely as they wish - this attribute simply defines the [MPD](#) validity duration.

Services may [publish in-band events to explicitly signal MPD validity](#) instead of expecting clients to regularly refresh on their own initiative. This enables finer control by the service but might not be supported by all clients.

This document requires:

- Services SHALL NOT require clients to support in-band events.

5.2.10. Timing of stand-alone IMSC1 and WebVTT text files⁵

Some services store [text adaptation sets](#) in stand-alone IMSC1 or WebVTT files, without segmentation or [\[ISOBMF F\]](#) encapsulation.

This document requires:

- Timecodes in stand-alone text files SHALL be relative to the [period](#) start point.
- `@presentationTimeOffset` SHALL NOT be present and SHALL be ignored by clients if present.

EXAMPLE 2

IMSC1 subtitles in stored in a stand-alone XML file.

```
<AdaptationSet mimeType="application/ttml+xml" lang="en-US">
  <Role schemeIdUri="urn:mpeg:dash:role:2011" value="subtitle" />
  <Representation>
    <BaseURL>subtitles_en_us.xml</BaseURL>
  </Representation>
</AdaptationSet>
```

Parts of the [MPD](#) structure that are not relevant for this chapter have been omitted - this is not a fully functional AdaptationSet element.

5.2.11. Forbidden techniques⁵

Some aspects of [\[MPEGDASH\]](#) are not compatible with the interoperable timing model defined in this document. In the interest of clarity, they are explicitly listed here:

- The `@presentationDuration` attribute SHALL NOT be used.

5.2.12. Examples⁵

This section is informative.

5.2.12.1. Offer content with imperfectly aligned tracks⁵

It may be that for various content processing workflow reasons, some tracks have a different duration from others. For example, the audio track might start a fraction of a second before the video track and end some time before the video track ends.

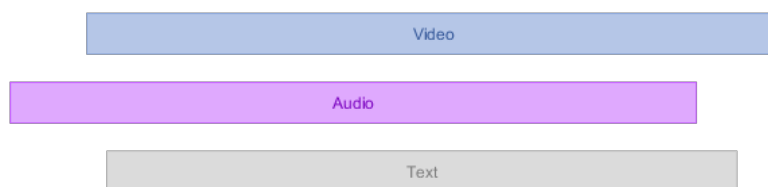


Figure 19 Content with different track lengths, before packaging as DASH.

You now have some choices to make in how you package these tracks into a DASH presentation that conforms to this document. Specifically, there exists the requirement that every [representation](#) must cover the entire [period](#) with media samples.

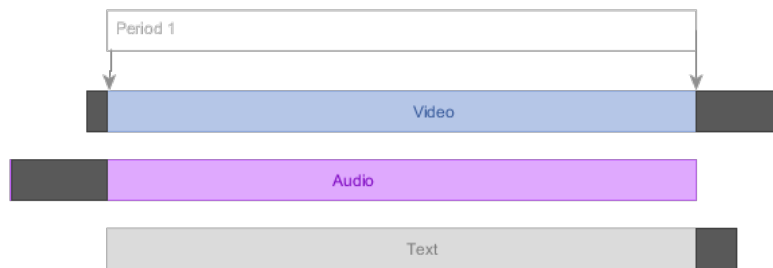


Figure 20 Content may be cut (indicated in black) to equalize track lengths.

The simplest option is to define a single [period](#) that contains [representations](#) resulting from cutting the content to match the shortest common time span, thereby covering the entire [period](#) with samples. Depending on the nature of the data that is removed, this may or may not be acceptable.



Figure 21 Content may be padded (indicated in green) to equalize track lengths.

If you wish to preserve track contents in their entirety, the most interoperable option is to add padding samples (e.g. silence or black frames) to all tracks to ensure that all [representations](#) have enough data to cover the entire [period](#) with samples. This may require customization of the encoding process, as the padding must match the codec configuration of the real content and might be impractical to add after the real content has already been encoded.

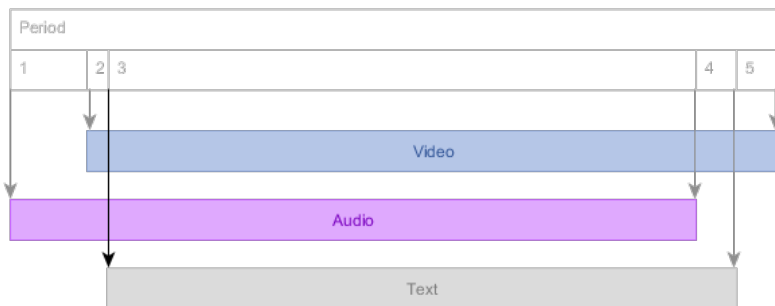


Figure 22 New [periods](#) may be started at any change in the set of available tracks.

Another option that preserves track contents is to [split the content](#) into multiple [periods](#) that each contain a different set of [representations](#), starting a new [period](#) whenever a track starts or ends. This enables you to ensure every [representations](#) covers its [period](#) with samples. The upside of this approach is that it can be done easily, requiring only manipulation of the MPD. The downside is that some clients may be unable to seamlessly play across every [period](#) transition.

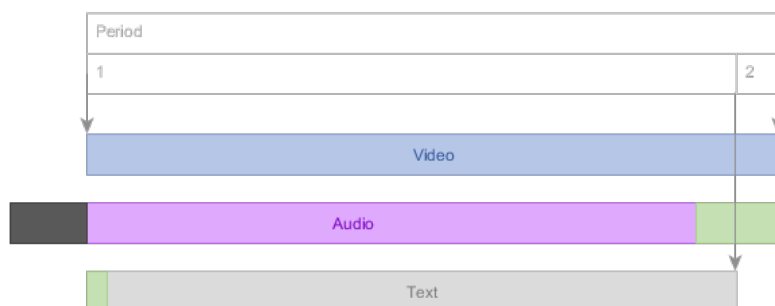


Figure 23 You may combine the different approaches, cutting in some places (black), padding in others (green) and defining multiple [periods](#) as needed.

You may wish to combine the different approaches, depending on the track, to achieve the optimal result.

Some clients are known to fail when transitioning from a [period](#) with audio and video to a [period](#) with only one of these components. You should avoid such transitions unless you have exact knowledge of the capabilities of your clients.

5.2.12.2. Split a period⁵

There exist scenarios where you would wish to split a [period](#) in two. Common reasons would be:

- to insert an ad [period](#) in the middle of an existing [period](#).
- parameters of one [adaptation set](#) change (e.g. KID or display aspect ratio), requiring a new [period](#) to update signaling.
- some [adaptation sets](#) become available or unavailable (e.g. different languages).

This example shows how an existing [period](#) can be split in a way that clients capable of [seamless period-connected playback](#) do not experience interruptions in playback among [representations](#) that are present both before and after the split.

Our starting point is a presentation with a single [period](#) that contains an audio [representation](#) with short samples and a video [representation](#) with slightly longer samples, so that [media segment](#) start points do not always overlap.

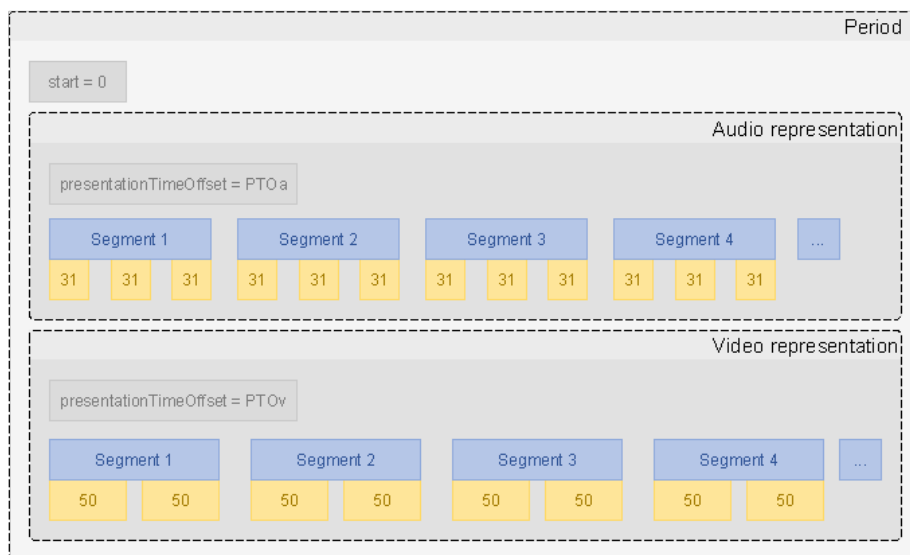


Figure 24 Presentation with one period, before splitting. Blue is a segment, yellow is a sample. Duration in arbitrary units is listed on samples. Segment durations are taken to be the sum of sample durations. `presentationTimeOffset` may have any value - it is listed because will be referenced later.

Note: [Periods](#) may be split at any point in time as long as both sides of the split remain in conformance to this document (e.g. each contains at least 1 [media segment](#)). Furthermore, [period](#) splitting does not require manipulation of the segments themselves, only manipulation of the MPD.

Let's split this [period](#) at position 220. This split occurs during segment 3 for both [representations](#) and during sample 8 and sample 5 of the audio and video [representation](#), respectively.

The mechanism that enables [period](#) splitting in the middle of a segment is the following:

- a [media segment](#) that overlaps a [period](#) boundary exists in both [periods](#).
- [representations](#) that are split are signaled in the MPD as [period-connected](#).
- a representation that is [period-connected](#) with a representation in a previous [period](#) [`#[#timing-connectivity]is` marked with the [period](#) connectivity descriptor].
- clients are expected to deduplicate boundary-overlapping [media segments](#) for [representations](#) on which [period connectivity](#) is signaled, if necessary for seamless playback (implementation-specific).
- clients are expected to present only the samples that are within the bounds of the current [period](#) (may be limited by client platform capabilities).

After splitting the example presentation, we arrive at the following structure.

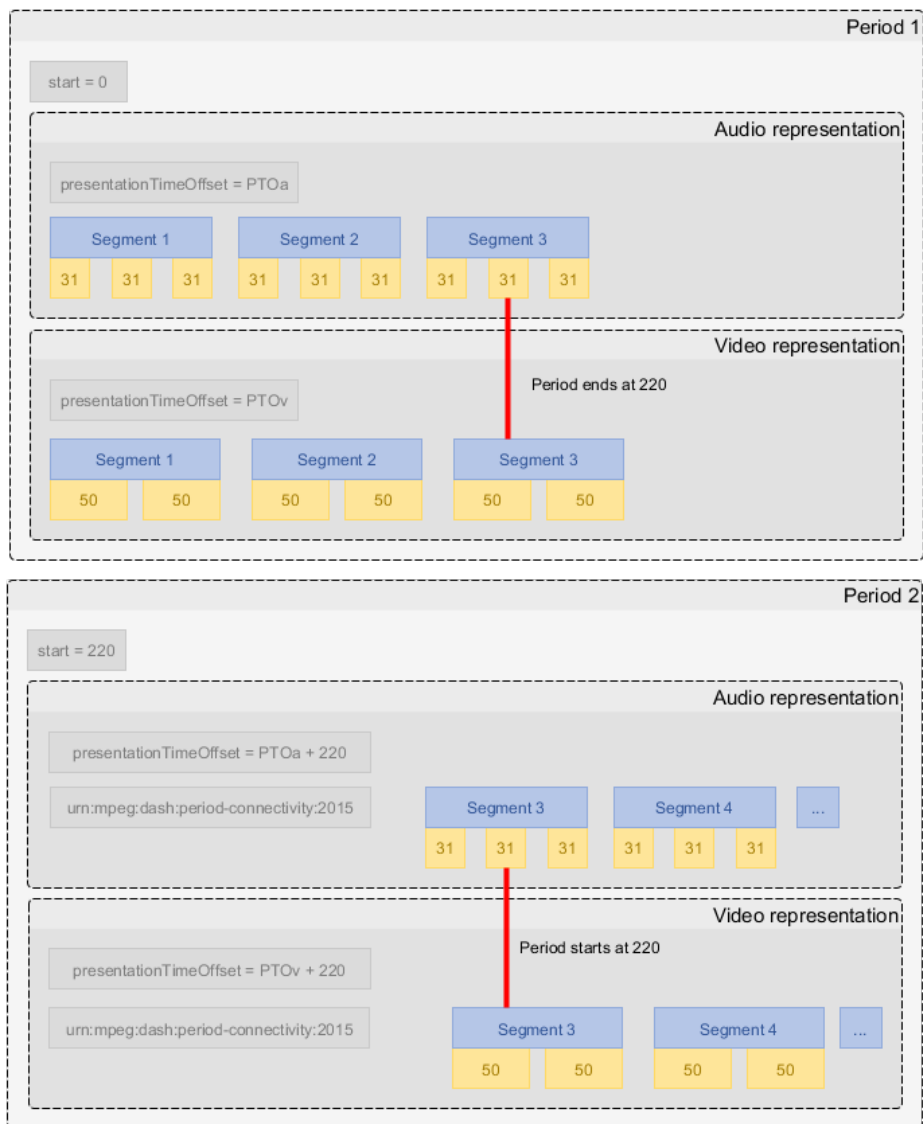


Figure 25 Presentation with two [periods](#), after splitting. Audio segment 3 and video segment 3 are shared by both [periods](#), with the connectivity signaling indicating that seamless playback with de-duplicating behavior is expected from clients.

If [indexed addressing](#) is used, both [periods](#) will reference all segments as both [periods](#) will use the same unmodified index segment. Clients are expected to ignore [media segments](#) that fall outside the [period](#) bounds.

Simple addressing has significant limitations on alignment at [period](#) start, making it unsuitable for some multi-period scenarios. See [§ 5.3.4.2 Moving the period start point \(simple addressing\)](#).

Other [periods](#) (e.g. ads) may be inserted between the two [periods](#) resulting from the split. This does not affect the addressing and timing of the two [periods](#).

5.2.12.3. Change the default_KID

In encrypted content, the `default_KID` of a [representation](#) might need to be changed at certain points in time. Often, the changes are closely synchronized in different [representations](#).

To perform the `default_KID` change, start a new [period](#) on every change, treating each [representation](#) as an independently changing element. With proper signaling, clients can perform this change seamlessly.

ISSUE 3 What about [period](#) connectivity? [#238](#)

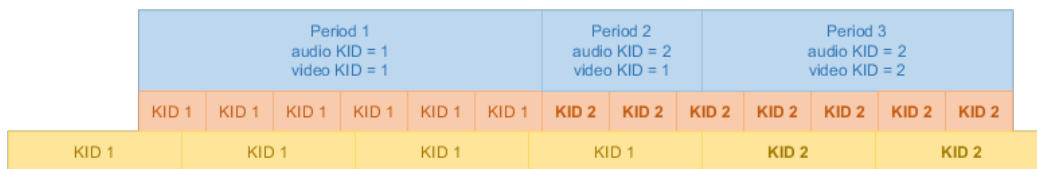


Figure 26 A change in `default_KID` starts a new period. Orange indicates audio and yellow video representation.

The same pattern can also be applied to other changes in representation configuration.

5.3. Segment addressing modes

This section defines the **addressing modes** that can be used for referencing media segments, initialization segments and index segments in interoperable DASH presentations.

Addressing modes not defined in this chapter SHALL NOT be used by DASH services. Clients SHOULD support all addressing modes defined in this chapter.

All representations in the same adaptation set SHALL use the same addressing mode. Representations in different adaptation sets MAY use different addressing modes. Period-connected representations SHALL use the same addressing mode in every period.

You SHOULD choose the addressing mode based on the nature of the content:

- ↪ **Content generated on the fly**
Use explicit addressing.
- ↪ **Content generated in advance of publishing**
Use indexed addressing or explicit addressing.

A service MAY use simple addressing which enables the packager logic to be very simple. This simplicity comes at a cost of reduced applicability to multi-period scenarios and reduced client compatibility.

Note: Future updates to [MPEGDASH] are expected to eliminate the critical limitations of simple addressing, enabling a wider range of applicable use cases.

ISSUE 4 Update to match [MPEGDASH] 4th edition.

Indexed addressing enables all data associated with a single representation to be stored in a single CMAF track file from which byte ranges are served to clients to supply media segments, the initialization segment and the index segment. This gives it some unique advantages:

- A single large file is more efficient to transfer and cache than 100 000 or more small files, reducing computational and I/O overhead.
- CDNs are aware of the nature of byte-range requests and can preemptively read-ahead to fill the cache ahead of playback.

5.3.1. Indexed addressing

A representation that uses **indexed addressing** consists of a CMAF track file containing an index segment, an initialization segment and a sequence of media segments.

Note: This addressing mode is sometimes called "SegmentBase" in other documents.

Clauses in section only apply to representations that use indexed addressing.

Note: [MPEGDASH] makes a distinction between "segment" (HTTP-addressable entity) and "subsegment" (byte range of an HTTP-addressable entity). This document does not make such a distinction and has no concept of subsegments. Usage of "segment" here matches the definition of CMAF segment [MPEGCMAF].

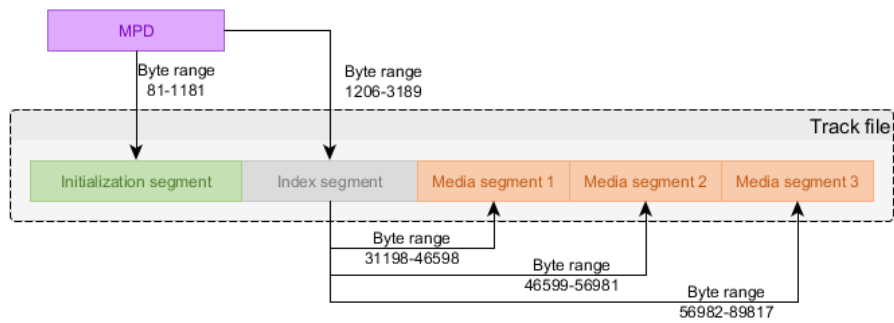


Figure 27 *Indexed addressing is based on an [index segment](#) that references all [media segments](#).*

The [MPD](#) defines the byte range in the [CMAF track file](#) that contains the [index segment](#). The [index segment](#) informs the client of all the [media segments](#) that exist, the time spans they cover on the [sample timeline](#) and their byte ranges.

Multiple [representations](#) SHALL NOT be stored in the same [CMAF track file](#) (i.e. no multiplexed [representations](#) are to be used).

At least one [Representation/BaseURL](#) element SHALL be present in the [MPD](#), containing a URL pointing to the [CMAF track file](#).

The [SegmentBase@indexRange](#) attribute SHALL be present in the [MPD](#). The value of this attribute identifies the byte range of the [index segment](#) in the [CMAF track file](#). The value is a [byte-range-spec](#) as defined in [\[RFC7233\]](#), referencing a single range of bytes.

The [SegmentBase@timescale](#) attribute SHALL be present and its value SHALL match the value of the [timescale](#) field in the [index segment](#) (in the [\[ISOBMFF\]](#) [sidx](#) box) and the value of the [timescale](#) field in the [initialization segment](#) (in the [\[ISOBMFF\]](#) [tkhd](#) box)).

The [SegmentBase/Initialization@range](#) attribute SHALL identify the byte range of the initialization segment in the [CMAF track file](#). The value is a [byte-range-spec](#) as defined in [\[RFC7233\]](#), referencing a single range of bytes. The [Initialization@sourceURL](#) attribute SHALL NOT be used.

EXAMPLE 3

Below is an example of common usage of [indexed addressing](#).

The example defines a [timescale](#) of 48000 units per second, with the [period](#) starting at position 8100 (or 0.16875 seconds) on the [sample timeline](#). The client can use the [index segment](#) referenced by [indexRange](#) to determine where the [media segment](#) containing position 8100 (and all other [media segments](#)) can be found. The byte range of the [initialization segment](#) is also provided.

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011">
  <Period>
    <AdaptationSet>
      <Representation>
        <BaseURL>showreel_audio_dashinit.mp4</BaseURL>
        <SegmentBase timescale="48000" presentationTimeOffset="8100" indexRange="848-999">
          <Initialization range="0-847"/>
        </SegmentBase>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Parts of the [MPD](#) structure that are not relevant for this chapter have been omitted - this is not a fully functional [MPD](#) file.

5.3.2. Structure of the index segment

The [index segment](#) SHALL consist of a single [Segment Index Box](#) ([sidx](#)) as defined by [\[ISOBMFF\]](#). The field layout is as follows:

```

aligned(8) class SegmentIndexBox extends FullBox('sidx', version, 0) {
    unsigned int(32) reference_ID;
    unsigned int(32) timescale;

    if (version==0) {
        unsigned int(32) earliest_presentation_time;
        unsigned int(32) first_offset;
    }
    else {
        unsigned int(64) earliest_presentation_time;
        unsigned int(64) first_offset;
    }

    unsigned int(16) reserved = 0;
    unsigned int(16) reference_count;

    for (i = 1; i <= reference_count; i++)
    {
        bit(1) reference_type;
        unsigned int(31) referenced_size;
        unsigned int(32) subsegment_duration;
        bit(1) starts_with_SAP;
        unsigned int(3) SAP_type;
        unsigned int(28) SAP_delta_time;
    }
}

```

The values of the fields are determined as follows:

reference_ID

The track_ID of the [\[ISOBMFF\]](#) track that contains the data of this [representation](#).

timescale

Same as the timescale field of the Media Header Box and same as the SegmentBase@timescale attribute in the [MPD](#).

earliest_presentation_time

The start timestamp of the first [media segment](#) on the [sample timeline](#), in [timescale units](#).

first_offset

Distance from the end of the [index segment](#) to the first [media segment](#), in bytes. For example, 0 indicates that the first [media segment](#) immediately follows the [index segment](#).

reference_count

Total number of [media segments](#) referenced by the [index segment](#).

reference_type

0

referenced_size

Size of the [media segment](#) in bytes. [Media segments](#) are assumed to be consecutive, so this is also the distance to the start of the next [media segment](#).

subsegment_duration

Duration of the [media segment](#) in [timescale units](#).

starts_with_SAP

1

SAP_type

Either 1 or 2, depending on the sample structure in the [media segment](#).

SAP_delta_time

0

ISSUE 5 We need to clarify how to determine the right value for SAP_type. [#235](#)

5.3.2.1. Moving the period start point (indexed addressing)⁵

When splitting [periods](#) in two or performing other types of editorial timing adjustments, a service might want to start a [period](#) at a point after the "natural" start point of the [representations](#) within.

For [representations](#) that use [indexed addressing](#), perform the following adjustments to set a new [period](#) start point:

1. Update SegmentBase@presentationTimeOffset to indicate the desired start point on the [sample timeline](#).

2. Update `Period@duration` to match the new duration.

5.3.3. Explicit addressing

A representation that uses **explicit addressing** consists of a set of [media segments](#) accessed via URLs constructed using a template defined in the [MPD](#), with the exact time span covered by each [media segment](#) described in the [MPD](#).

Note: This addressing mode is sometimes called "SegmentTemplate with SegmentTimeline" in other documents.

Clauses in section only apply to [representations](#) that use [explicit addressing](#).

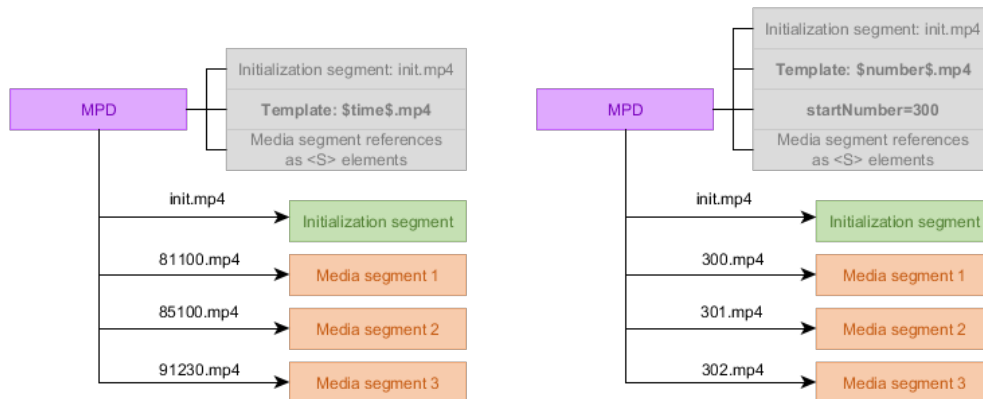


Figure 28 [Explicit addressing](#) uses a segment template that is combined with explicitly defined time spans for each [media segment](#) in order to reference [media segments](#), either by start time or by sequence number.

The [MPD](#) SHALL contain a `SegmentTemplate/SegmentTimeline` element, containing a set of [segment references](#), which satisfies the requirements defined in this document. The [segment references](#) exist as a sequence of `S` elements, each of which references one or more [media segments](#) with start time `S@t` and duration `S@d timescale units` on the [sample timeline](#). The `SegmentTemplate@duration` attribute SHALL NOT be present.

To enable concise [segment reference](#) definitions, an `S` element may represent a repeating [segment reference](#) that indicates a number of repeated consecutive [media segments](#) with the same duration. The value of `S@r` SHALL indicate the number of additional consecutive [media segments](#) that exist.

Note: Only additional [segment references](#) are counted, so `S@r=5` indicates a total of 6 consecutive [media segments](#) with the same duration.

The start time of a [media segment](#) is calculated from the start time and duration of the previous [media segment](#) if not specified by `S@t`. There SHALL NOT be any gaps or overlap between [media segments](#).

The value of `S@r` is nonnegative, except for the last `S` element which MAY have a negative value in `S@r`, indicating that the repeated [segment references](#) continue indefinitely up to a [media segment](#) that either ends at or overlaps the [period](#) end point.

[Updates to a dynamic MPD](#) MAY add more `S` elements, remove expired `S` elements, increment `SegmentTemplate@startNumber`, add the `S@t` attribute to the first `S` element or increase the value of `S@r` on the last `S` element but SHALL NOT otherwise modify existing `S` elements.

The `SegmentTemplate@media` attribute SHALL contain the URL template for referencing [media segments](#), using either the `$Time$` or `$Number$` template variable to uniquely identify [media segments](#). The `SegmentTemplate@initialization` attribute SHALL contain the URL template for referencing [initialization segments](#).

If using `$Number$` addressing, the number of the first segment reference is defined by `SegmentTemplate@startNumber` (default value 1). The `S@n` attribute SHALL NOT be used - segment numbers form a continuous sequence starting with `SegmentTemplate@startNumber`.

EXAMPLE 4

Below is an example of common usage of [explicit addressing](#).

The example defines 225 [media segments](#) starting at position 900 on the [sample timeline](#) and lasting for a total of 900.225 seconds. The [period](#) ends at 900 seconds, so the last 0.225 seconds of content is clipped (out of bounds samples may also simply be omitted from the last [media segment](#)). The [period](#) starts at position 900 which matches the start position of the first [media segment](#) found at the relative URL `video/900.m4s`.

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011">
  <Period duration="PT900S">
    <AdaptationSet>
      <Representation>
        <SegmentTemplate timescale="1000" presentationTimeOffset="900"
          media="video/$Time$.m4s" initialization="video/init.mp4">
          <SegmentTimeline>
            <S t="900" d="4001" r="224" />
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Parts of the [MPD](#) structure that are not relevant for this chapter have been omitted - this is not a fully functional [MPD](#) file.

EXAMPLE 5

Below is an example of [explicit addressing](#) used in a scenario where different [media segments](#) have different durations (e.g. due to encoder limitations).

The example defines a sequence of 11 [media segments](#) starting at position 120 on the [sample timeline](#) and lasting for a total of 95520 units at a [timescale](#) of 1000 units per second (which results in 95.52 seconds of data). The [period](#) starts at position 810, which is within the first [media segment](#), found at the relative URL `video/120.m4s`. The fifth [media segment](#) repeats once, resulting in a sixth [media segment](#) with the same duration.

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011">
  <Period>
    <AdaptationSet>
      <Representation>
        <SegmentTemplate timescale="1000" presentationTimeOffset="810"
          media="video/$Time$.m4s" initialization="video/init.mp4">
          <SegmentTimeline>
            <S t="120" d="8520"/>
            <S d="8640"/>
            <S d="8600"/>
            <S d="8680"/>
            <S d="9360" r="1"/>
            <S d="8480"/>
            <S d="9080"/>
            <S d="6440"/>
            <S d="10000"/>
            <S d="8360"/>
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Parts of the [MPD](#) structure that are not relevant for this chapter have been omitted - this is not a fully functional [MPD](#) file.

5.3.3.1. Moving the period start point (explicit addressing)

When splitting [periods](#) in two or performing other types of editorial timing adjustments, a service might want to start a

[period](#) at a point after the "natural" start point of the [representations](#) within.

For [representations](#) that use [explicit addressing](#), perform the following adjustments to set a new [period](#) start point:

1. Update `SegmentTemplate@presentationTimeOffset` to indicate the desired start point on the [sample timeline](#).
2. Update `Period@duration` to match the new duration.
3. Remove any [unnecessary segment references](#).
4. If using the `$Number$` template variable, increment `SegmentTemplate@startNumber` by the number of [media segments](#) removed from the beginning of the [representation](#).

Note: See [§ 5.2.4 Representations](#) and [§ 5.2.9.5.2 Removing content from the MPD](#) to understand the constraints that apply to [segment reference](#) removal.

5.3.4. Simple addressing

ISSUE 6 Once we have a specific `@earliestPresentationTime` proposal submitted to MPEG we need to update this section to match. See [#245](#). This is now done in [\[MPEGDASH\]](#) 4th edition - need to synchronize this text.

A representation that uses **simple addressing** consists of a set of [media segments](#) accessed via URLs constructed using a template defined in the [MPD](#), with the nominal time span covered by each [media segment](#) described in the [MPD](#).

Simple addressing defines the nominal time span of each [media segment](#) in the [MPD](#). The true time span covered by samples within the [media segment](#) can be slightly different than the nominal time span. See [§ 5.3.4.1 Inaccuracy in media segment timing when using simple addressing](#).

Note: This addressing mode is sometimes called "SegmentTemplate without SegmentTimeline" in other documents.

Clauses in section only apply to [representations](#) that use [simple addressing](#).

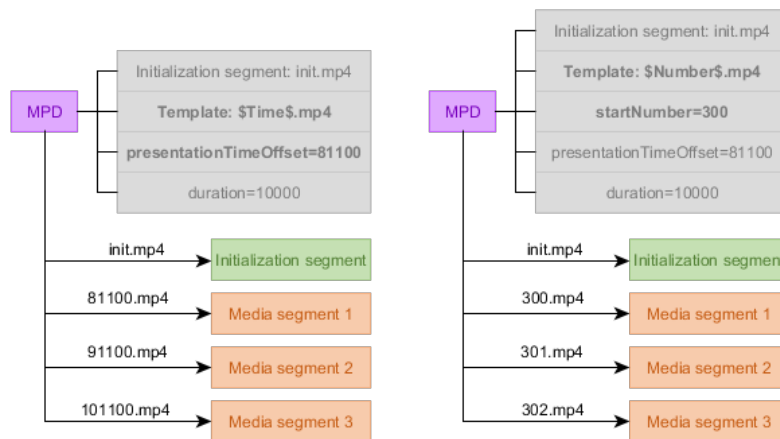


Figure 29 Simple addressing uses a segment template that is combined with approximate first [media segment](#) timing information and an average [media segment](#) duration in order to reference [media segments](#), either by start time or by sequence number.

The `SegmentTemplate@duration` attribute SHALL define the nominal duration of a [media segment](#) in [timescale units](#).

The set of [segment references](#) SHALL consist of the first [media segment](#) starting exactly at the [period](#) start point and all other [media segments](#) following in a consecutive series of equal time spans of `SegmentTemplate@duration` [timescale units](#), ending with a [media segment](#) that ends at or overlaps the [period](#) end time.

The `SegmentTemplate@media` attribute SHALL contain the URL template for referencing [media segments](#), using either the `$Time$` or `$Number$` template variable to uniquely identify [media segments](#). The `SegmentTemplate@initialization` attribute SHALL contain the URL template for referencing initialization segments.

If using `$Number$` addressing, the number of the first segment reference is defined by `SegmentTemplate@startNumber` (default value 1).

EXAMPLE 6

Below is an example of common usage of [simple addressing](#).

The example defines a [sample timeline](#) with a [timescale](#) of 1000 units per second, with the [period](#) starting at position 900. The average duration of a [media segment](#) is 4001. [Media segment](#) numbering starts at 800, so the first [media segment](#) is found at the relative URL `video/800.m4s`. The sequence of [media segments](#) continues to the end of the period, which is 900 seconds long, making for a total of 225 defined [segment references](#).

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011">
  <Period duration="PT900S">
    <AdaptationSet>
      <Representation>
        <SegmentTemplate timescale="1000" presentationTimeOffset="900"
          media="video/$Number$.m4s" initialization="video/init.mp4"
          duration="4001" startNumber="800" />
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Parts of the [MPD](#) structure that are not relevant for this chapter have been omitted - this is not a fully functional [MPD](#) file.

5.3.4.1. Inaccuracy in media segment timing when using simple addressing

When using [simple addressing](#), the samples contained in a [media segment](#) MAY cover a different time span on the [sample timeline](#) than what is indicated by the nominal timing in the [MPD](#), as long as no constraints defined in this document are violated by this deviation.

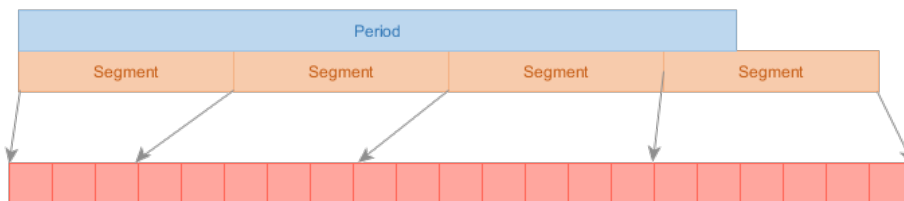


Figure 30 [Simple addressing](#) relaxes the requirement on [media segment](#) contents matching the [sample timeline](#). Red boxes indicate samples.

The allowed deviation is defined as the maximum offset between the edges of the nominal time span (as defined by the [MPD](#)) and the edges of the true time span (as defined by the contents of the [media segment](#)). The deviation is evaluated separately for each edge.

This allowed deviation does not relax any requirements that do not explicitly define an exception. For example, [periods](#) must still be covered with samples for their entire duration, which constrains the flexibility allowed for the first and last [media segment](#) in a [period](#).

The deviation SHALL be no more than 50% of the nominal [media segment](#) duration and MAY be in either direction.

Note: This results in a maximum true duration of 200% (+50% outward extension on both edges) and a minimum true duration of 1 sample (-50% inward from both edges would result in 0 duration but empty [media segments](#) are not allowed).

Allowing inaccurate timing is intended to enable reasoning on the [sample timeline](#) using average values for [media segment](#) timing. If the addressing data says that a [media segment](#) contains 4 seconds of data on average, a client can predict with reasonable accuracy which samples are found in which [media segments](#), while at the same time the service is not required to publish per-segment timing data in the [MPD](#). It is expected that the content is packaged with this constraint in mind (i.e. **every** segment cannot be inaccurate in the same direction - a shorter segment now implies a longer segment in the future to make up for it).

EXAMPLE 7

Consider a [media segment](#) with a nominal start time of 8 seconds from [period](#) start and a nominal duration of 4 seconds, within a [period](#) of unlimited duration.

The following are all valid contents for such a [media segment](#):

- samples from 8 to 12 seconds (perfect accuracy)
- samples from 6 to 14 seconds (maximally large segment allowed, 50% increase from both ends)
- samples from 9.9 to 10 seconds (near-minimally small segment; while we allow a 50% decrease from both ends, potentially resulting in zero duration, every segment must still contain at least one sample)
- samples from 6 to 10 seconds (maximal offset toward zero point at both ends)
- samples from 10 to 14 seconds (maximal offset away from zero point at both ends)

Near [period](#) boundaries, all the constraints of timing and addressing must still be respected! Consider a [media segment](#) with a nominal start time of 0 seconds from [period](#) start and a nominal duration of 4 seconds. If such a [media segment](#) contained samples from 1 to 5 seconds (offset of 1 second away from zero point at both ends, which is within acceptable limits) it would be non-conforming because of the requirement in [§ 5.2.7 Media segments](#) that the first [media segment](#) contain a media sample that starts at or overlaps the [period](#) start point. This severely limits the usefulness of [simple addressing](#).

5.3.4.2. Moving the period start point (simple addressing)

When splitting [periods](#) in two or performing other types of editorial timing adjustments, a service might want to start a [period](#) at a point after the "natural" start point of the [representations](#) within.

[Simple addressing](#) is challenging to use in such scenarios. You SHOULD convert [simple addressing representations](#) to use [explicit addressing](#) before adjusting the [period](#) start point or splitting a [period](#). See [§ 5.3.4.3 Converting simple addressing to explicit addressing](#).

The rest of this chapter provides instructions for situations where you choose **not** to convert to [explicit addressing](#).

To move the [period](#) start point, for [representations](#) that use [simple addressing](#):

- Every [simple addressing representation](#) in the [period](#) must contain a [media segment](#) that starts exactly at the new [period](#) start point.
- [Media segments](#) starting at the new [period](#) start point must contain a sample that starts at or overlaps the new [period](#) start point.

Note: If you are splitting a [period](#), also keep in mind [the requirements on period end point sample alignment](#) for the [period](#) that remains before the split point.

Finding a suitable new start point that conforms to the above requirements can be very difficult. If inaccurate timing is used, it may be altogether impossible. This is a limitation of [simple addressing](#).

Having ensured conformance to the above requirements for the new [period](#) start point, perform the following adjustments:

1. Update `SegmentTemplate@presentationTimeOffset` to indicate the desired start point on the [sample timeline](#).
2. If using the `$Number$` template variable, increment `SegmentTemplate@startNumber` by the number of [media segments](#) removed from the beginning of the [representation](#).
3. Update `Period@duration` to match the new duration.

5.3.4.3. Converting simple addressing to explicit addressing

It may sometimes be desirable to convert a presentation from [simple addressing](#) to [explicit addressing](#). This chapter provides an algorithm to do this.

[Simple addressing](#) allows for inaccuracy in [media segment](#) timing. No inaccuracy is allowed by [explicit addressing](#). The mechanism of conversion described here is only valid when there is no inaccuracy. If the nominal time spans in original the [MPD](#) differ from the true time spans of the [media segments](#), re-package the content from scratch using [explicit addressing](#) instead of converting.

To perform the conversion, execute the following steps:

1. Calculate the number of [media segments](#) in the [representation](#) as $\text{SegmentCount} = \text{Ceil}(\text{AsSeconds}(\text{Period@duration}) / (\text{SegmentTemplate@duration} / \text{SegmentTemplate@timescale}))$.
2. Update the MPD.
 1. Add a single `SegmentTemplate/SegmentTimeline` element.
 2. Add a single `SegmentTimeline/S` element.
 3. Set `S@t` to equal `SegmentTemplate@presentationTimeOffset`.
 4. Set `S@d` to equal `SegmentTemplate@duration`.
 5. Remove `SegmentTemplate@duration`.
 6. Set `S@r` to `SegmentCount - 1`.

EXAMPLE 8

Below is an example of a [simple addressing representation](#) before conversion.

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011">
  <Period duration="PT900S">
    <AdaptationSet>
      <Representation>
        <SegmentTemplate timescale="1000" presentationTimeOffset="900"
          media="video/$Number$.m4s" initialization="video/init.mp4"
          duration="4001" startNumber="800" />
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

As part of the conversion, we calculate $\text{SegmentCount} = \text{Ceil}(900 / (4001 / 1000)) = 225$.

After conversion, we arrive at the following result.

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011">
  <Period duration="PT900S">
    <AdaptationSet>
      <Representation>
        <SegmentTemplate timescale="1000" presentationTimeOffset="900"
          media="video/$Number$.m4s" initialization="video/init.mp4"
          startNumber="800">
          <SegmentTimeline>
            <S t="900" d="4001" r="224" />
          </SegmentTimeline>
        </SegmentTemplate>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Parts of the [MPD](#) structure that are not relevant for this chapter have been omitted - the above are not fully functional [MPD](#) files.

5.4. Adaptation set contents

[Adaptation sets](#) SHALL contain [media segments](#) compatible with a single decoder, although services MAY require the decoder to be re-initialized when switching to a new [representation](#). See also [§ 6.4 Bitstream switching](#).

All [representations](#) in the same [adaptation set](#) SHALL have the same [timescale](#), both in the [MPD](#) and in the [initialization segment](#) `tkhd` boxes.

[\[ISOBMFF\]](#) edit lists SHALL be identical for all [representations](#) in an [adaptation set](#).

Note: [\[DVB-DASH\]](#) defines some relevant constraints in section 4.5. Consider obeying these constraints to be compatible with [\[DVB-DASH\]](#).

5.5. Adaptation set types§

Each [adaptation set](#) SHALL match exactly one category from among the following:

- A **video adaptation set** contains visual information for display to the user. Such an adaptation set is identified by `@mimeType="video/mp4"`. The values for `@codecs` SHALL be restricted to values defined in [§ 11 Media coding technologies](#).
- An **audio adaptation set** contains sound information to be rendered to the user. Such an adaptation set is identified by `@mimeType="audio/mp4"`. The values for `@codecs` SHALL be restricted to values defined in [§ 11 Media coding technologies](#).
- A **text adaptation set** contains visual overlay information to be rendered as auxiliary or accessibility information. Such an [adaptation set](#) is identified by one of:
 - `@mimeType="application/mp4"` and a `@codecs` parameter of a text coding technology defined in [§ 11 Media coding technologies](#).
 - `@mimeType="application/ttml+xml"` with no `@codecs` parameter.
- A metadata adaptation set contains information that is not expected to be rendered by a specific media handler, but is interpreted by the application. Such an adaptation set is identified by `@mimeType="application/mp4"` and an appropriate sample entry identified by the `@codecs` parameter.
- A **thumbnail adaptation set** contains [thumbnail images for efficient display during seeking](#). Such an adaptation set is identified by `@mimeType="image/jpeg"` or `@mimeType="image/png"` in combination with an [essential property descriptor](#) with `@schemeIdUri="http://dashif.org/guidelines/thumbnail_tile"`.

ISSUE 7 What exactly is metadata `@codecs` supposed to be? <https://github.com/Dash-Industry-Forum/DASH-IF-IOP/issues/290>

The [adaptation set](#) type SHALL be used by a DASH client to identify the appropriate handler for rendering. Typically, a DASH client selects at most one [adaptation set](#) of each type.

In addition, a DASH client SHOULD use the value of the `@codecs` parameter to determine whether the underlying media playback platform can play the media contained within the [adaptation set](#).

See [§ 11 Media coding technologies](#) for detailed codec-specific constraints.

5.6. Video adaptation set constraints§

All [representations](#) in the same [video adaptation set](#) SHALL be alternative encodings of the same source content, encoded such that switching between them does not produce visual glitches due to picture size or aspect ratio differences.

ISSUE 8 An illustration here would be very useful.

ISSUE 9 <https://github.com/Dash-Industry-Forum/DASH-IF-IOP/issues/284>

To avoid visual glitches you must ensure that the sample aspect ratio is set correctly. For reasons of coding efficiency and due to technical constraints, different [representations](#) might use a different picture aspect ratio. Each [representation](#) signals a sample aspect ratio (e.g. in an [\[MPEGAVC\]](#) `aspect_ratio_idc`) that is used to scale the picture so that every [representation](#) ends up at the same display aspect ratio. The formula is `display aspect ratio = picture aspect ratio / sample aspect ratio`.

In the [MPD](#), the display aspect ratio is `AdaptationSet@par` and the sample aspect ratio is `Representation@sar`. The picture aspect ratio is not directly present but is derived from `Representation@width` and `Representation@height`.

The encoded picture SHALL only contain the active video area, so that clients 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".

[Representations](#) in the same [video adaptation set](#) SHALL NOT differ in any of the following parameters:

- Color Primaries
- Transfer Characteristics
- Matrix Coefficients.

If different [video adaptation sets](#) differ in any of the above parameters, these parameters SHOULD be signaled in the MPD on the [adaptation set](#) level by a [supplemental property descriptor](#) or an [essential property descriptor](#) with `@schemeIdUri="urn:mpeg:mpegB:cicp:<Parameter>"` as defined in [\[iso23001-8\]](#) and `<Parameter>` being one of the following: `ColourPrimaries`, `TransferCharacteristics`, or `MatrixCoefficients`. The `@value` attribute SHALL be set as defined in [\[iso23001-8\]](#).

ISSUE 10 Why is the above a SHOULD? If it matters enough to signal, we should make it SHALL?
<https://github.com/Dash-Industry-Forum/DASH-IF-IOP/issues/286>

In any [video adaptation set](#), the following SHALL be present:

- `AdaptationSet@par` (the display aspect ratio)
- `Representation@sar` (the sample aspect ratio)
- Either `Representation@width` or `AdaptationSet@width` (but not both)
- Either `Representation@height` or `AdaptationSet@height` (but not both)
- Either `Representation@frameRate` or `AdaptationSet@frameRate` (but not both)

Note: `@width` and `@height` indicate the number of encoded pixels. `@par` indicates the final intended display aspect ratio and `@sar` is effectively the ratio of aspect ratios (ratio of `@width` x `@height` to `@par`).

EXAMPLE 9

Given a coded picture of 720x576 pixels with an intended display aspect ratio of 16:9, we would have the following values:

- `@width=720`
- `@height=576`
- `@par=16:9`
- `@sar=45:64` (720x576 is 5:4, which gives `@sar=5:4/16:9=45:64`)

ISSUE 11 This chapter already includes changes from [#274](#)

In any [video adaptation set](#), the following SHOULD NOT be present and SHALL be ignored by clients if present:

- `AdaptationSet@minWidth`
- `AdaptationSet@maxWidth`
- `AdaptationSet@minHeight`
- `AdaptationSet@maxHeight`
- `AdaptationSet@minFrameRate`
- `AdaptationSet@maxFrameRate`

The above min/max values are trivial to determine at runtime, so can be calculated by the client when needed.

`@scanType` SHOULD NOT be present and if present SHALL have the value `progressive`. Non-progressive video is not interoperable.

5.7. Audio adaptation set constraints

`AdaptationSet@lang` SHALL be present on every [audio adaptation set](#).

`@audioSamplingRate` SHALL be present either on the [adaptation set](#) or [representation](#) level (but not both).

The `AudioChannelConfiguration` element SHALL be present either on the [adaptation set](#) or [representation](#) level (but not both). The scheme and value SHALL conform to `ChannelConfiguration` as defined in [\[iso23001-8\]](#).

5.8. Text adaptation set constraints

[Text adaptation sets](#) SHOULD be annotated using descriptors defined by [\[MPEGDASH\]](#), specifically `Role`, `Accessibility`, `EssentialProperty` and `SupplementalProperty` descriptors.

Guidelines for annotation are provided in [§ 7 Content annotation and selection](#) and section 7.1.2 of [\[DVB-DASH\]](#).

5.9. Accessing resources over HTTP§

[MPEGDASH] defines the structure of DASH presentations. Combined with an understanding of the [addressing modes], this enables DASH clients to determine a set of HTTP requests that must be made to acquire the resources needed for playback of a DASH presentation. This section defines rules for performing the HTTP requests and signaling the relevant parameters in an interoperable manner.

ISSUE 12 <https://github.com/Dash-Industry-Forum/DASH-IF-IOP/issues/333>

5.9.1. MPD URL resolution§

A service MAY use the `MPD/Location` element to redirect clients to a different URL to perform [MPD refreshes](#). HTTP redirection MAY be used when responding to client requests.

A DASH client performing an [MPD refresh](#) SHALL determine the MPD URL according to the following algorithm:

1. If at least one `MPD/Location` element is present, the value of any `MPD/Location` element is used as the MPD URL. Otherwise the original MPD URL is used as the MPD URL.
2. If the HTTP request results in an HTTP redirect using a 3xx response code, the redirected URL replaces the MPD URL.

The MPD URL as defined by the above algorithm SHALL be used as an implicit base URL for [media segment](#) requests.

Any present `BaseURL` element SHALL NOT affect MPD location resolution.

5.9.2. Segment URL resolution§

A service MAY publish [media segments](#) on URLs unrelated to the [MPD](#) URL. A service MAY use multiple `BaseURL` elements on any level of the MPD to offer content on multiple URLs (e.g. via multiple CDNs). HTTP redirection MAY be used when responding to client requests.

For [media segment](#) requests, the DASH client SHALL determine the URL according to the following algorithm:

1. If an absolute [media segment](#) URL is present in the MPD, it is used as-is (after [template variable substitution](#), if appropriate).
2. If an absolute `BaseURL` element is present in the MPD, it is used as the base URL.
3. Otherwise the MPD URL is used as the base URL, taking into account any MPD URL updates that occurred due to [MPD refreshes](#).
4. The base URL is combined with the relative [media segment](#) URL.

Note: The client may use any logic to determine which `BaseURL` to use if multiple are provided.

The same logic SHALL be used for [initialization segments](#) and [index segments](#).

ISSUE 13 What do relative BaseURLs do? Do they just incrementally build up the URL? Or are they ignored? This algorithm leaves it unclear, only referencing absolute BaseURLs. We should make it explicit.

5.9.3. Conditional MPD downloads§

It can often be the case that a [live service](#) signals a short [MPD](#) validity period to allow for the possibility of terminating the last [period](#) with minimal end-to-end latency. At the same time, generating future [segment references](#) might not require any additional information to be obtained by clients. That is, a situation might occur where constant [MPD refreshes](#) are required but the [MPD](#) content rarely changes.

Clients using HTTP to perform [MPD refreshes](#) SHOULD use conditional GET requests as specified in [RFC7232] to avoid unnecessary data transfers when the contents of the [MPD](#) do not change between refreshes.

5.9.4. Expanding URL template variables§

This section clarifies expansion rules for URL template variables such as `$Time$` and `$Number`, defined by [\[MPEGDASH\]](#).

The set of string formatting suffixes used SHALL be restricted to `%0[width]d`.

Note: The string format suffixes are not intended for general-purpose string formatting. Restricting it to only this single suffix enables the functionality to be implemented without a string formatting library.

5.10. Minimum buffer time signaling

ISSUE 14 The text here is technically correct but could benefit from being reworded in a simpler and more understandable way. If anyone finds themselves with the time, an extra pass over this would be helpful.

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 [media segment](#) duration****. The MBT may be set to a smaller value than maximum [media 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 [media segment](#) granularity. As in this case, not only parts of the [media segment](#) (i.e., MBT worth of data) needs to be downloaded, but the entire Segment, and if the MBT is smaller than the [media segment](#) duration, then rather the [media 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 $BW[i] \leq rate * buffer / max_segment_duration$.

5.11. Large timescales and time values

[\[ECMAScript\]](#) is unable to accurately represent numeric values greater than 2^{53} using built-in types. Therefore, interoperable services cannot use such values.

All timescales are start times used in a DASH presentations SHALL be sufficiently small that no timecode value exceeding 2^{53} will be encountered, even during the publishing of long-lasting [live services](#).

Note: This may require the use of 64-bit fields, although the values must still be limited to under 2^{53} .

5.12. MPD size§

No constraints are defined on [MPD](#) size, or on the number of elements. However, services SHOULD NOT create unnecessarily large [MPDs](#).

Note: [\[DVB-DASH\]](#) defines some relevant constraints in section 4.5. Consider obeying these constraints to be compatible with [\[\[DVB DASH\]\]](#).

5.13. Representing durations in XML§

All units expressed in [MPD](#) fields of datatype `xs:duration` SHALL be treated as fixed size:

- 60S = 1M (minute)
- 60M = 1H
- 24H = 1D
- 30D = 1M (month)
- 12M = 1Y

[MPD](#) fields having datatype `xs:duration` SHALL NOT use the year and month units and SHOULD be expressed as a count of seconds, without using any of the larger units.

6. Commonly used features§

This chapter describes some features of DASH presentations in their common implementations.

Not every DASH client will support each of these features. Compatibility of different clients and services can be verified by comparing the feature sets supported by clients and used by services (and may require experimentation and testing).

6.1. Seamless switching§

A key feature of DASH is the ability for clients to seamlessly switch between compatible [representations](#) at predetermined points on the [MPD timeline](#), enabling content from different [representations](#) to be interleaved according to the wishes of the client. This enables adaptive streaming - changing the active quality level in accordance with dynamically changing network conditions. Most DASH presentations define switching points at 1-10 second intervals.

Note: Decoder reinitialization during [representation](#) switches may result in visible or audible artifacts on some clients.

There are IDR-like SAPs (i.e. SAPs of type 1 or 2) at the start of each [media segment](#). This enables seamless switching. The presence of such SAPs is signaled in the [MPD](#) by providing a value of 1 or 2, depending on the sample structure of the [media segments](#), for either `AdaptationSet@subsegmentStartsWithSAP` (if [indexed addressing](#) is used) or `AdaptationSet@segmentStartsWithSAP` (if any other [addressing mode](#) is used).

ISSUE 15 We need to clarify how to determine the right value for `startsWithSAP`. [#235](#)

ISSUE 16 Add a reference here to help readers understand what are "IDR-like SAPs (i.e. SAPs of type 1 or 2)".

See also [§ 6.4 Bitstream switching](#).

6.2. Preview thumbnails for seeking and navigation§

Clients may wish to show timecode-associated preview thumbnails as part of the seeking experience. A typical use case is for enhancing a scrub bar with visual cues. Services that wish to support this SHOULD provide an [adaptation set](#) with thumbnails.

The thumbnails are published as a sequence of jpeg/png images containing grids of thumbnails. One grid of thumbnails is one [media segment](#). To ensure efficient transfer, a thumbnail [media segment](#) SHOULD be at least 1 minute in duration.

A [thumbnail adaptation set](#) MAY offer multiple [representations](#) with different spatial resolutions.

The [addressing mode](#) SHALL be restricted to [simple addressing](#) with only the `Number` templating variable.

Note: The constraint on allowed [addressing modes](#) exists to limit the effort required to implement this feature in clients.

Detailed requirements on the thumbnail [representations](#) are defined in [§ 11.5 Thumbnail images](#).

6.3. Trick mode§

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, the service SHOULD be structured as follows:

- add [adaptation sets](#) that only contain trick mode [representations](#)
- annotate each [adaptation set](#) with an [essential property descriptor](#) or [supplemental property descriptor](#) with URL <http://dashif.org/guidelines/trickmode> and the `value` the value of `id` attribute of the [adaptation set](#) with which these trick mode [representations](#) are associated. The trick mode [representations](#) must be time-aligned with the [representations](#) in the referenced [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 the [representation](#) is encoded without any coding dependency on the elementary stream level, i.e. each sample is a SAP type 1, then you SHOULD set the `Representation@codingDependency` attribute to `false`.
- If multiple trick mode [adaptation sets](#) are present for one regular [adaptation set](#), then sufficient signaling should be provided to differentiate the different trick mode [adaptation sets](#). Different [adaptation sets](#) for example may be provided as thumbnails (low spatial resolution), for fast forward or rewind (no coding dependency with `codingDependency` set to `false` and/or lower frame rates), longer values for `duration` to improve download frequencies or different `maxPlayoutRate` values. Note also that the `bandwidth` value should be carefully documented to support faster than real-time download of Segments.

If an [adaptation set](#) is annotated with the [essential property descriptor](#) with URI <http://dashif.org/guidelines/trickmode> then the DASH client SHALL NOT select any of the contained [representations](#) for regular playout.

6.4. Bitstream switching§

Bitstream switching is a feature that allows a switched sequence of [media segments](#) from different [representations](#) in the same [adaptation set](#) to be decoded without resetting the decoder at switch points by ensuring that the resulting stream of [media segments](#) can be successfully decoded without the decoder even being aware of a switch.

An [adaptation set](#) that supports bitstream switching is a **bitstream switching adaptation set**.


The `AdaptationSet@bitstreamSwitching` attribute SHOULD be set to `true` on a [bitstream switching adaptation set](#). Services SHALL NOT require clients to support bitstream switching in order to correctly present a [bitstream switching adaptation set](#).

The `[ISOBMFF] track_id` SHALL be equal for all [representations](#) in the same [bitstream switching adaptation set](#).

The `AdaptationSet@codecs` attribute SHALL be present on a [bitstream switching adaptation set](#) and indicate the maximum profile and level of any [representation](#).

The `Representation@codecs` attribute MAY be present on [representations](#) that belong to a [bitstream switching adaptation set](#). If present, it SHALL indicate the maximum profile and level of any [media segment](#) in the

[representation](#).

ISSUE 17  Allowing `Representation@codecs` to be absent might make it more difficult to make bitstream-switching-oblivious clients. If we require `Representation@codecs` to always be present, client developer life could be made simpler.

Clients that support bitstream switching SHALL initialize the decoder using the [initialization segment](#) of the [representation](#) with the highest `Representation@bandwidth` in a bitstream switching [adaptation set](#).

Note: A [bitstream switching adaptation set](#) fulfills the requirements of [\[DVB-DASH\]](#).

6.5. Switching across adaptation sets[§]

Note: This technology is expected to be available in [\[MPEGDASH\]](#) Amd 4. Once published by MPEG, this section is expected to be replaced by a reference to the MPEG-DASH standard.

Representations in two or more [adaptation sets](#) may provide the same content. In addition, the content may be time-aligned and may be offered such that seamless switching across [representations](#) in different [adaptation sets](#) is possible. Typical examples are the offering of the same content with different codecs, for example H.264/AVC and H.265/HEVC and the content author wants to provide such information to the receiver in order to seamlessly switch [representations](#) across different [adaptation sets](#). Such switching permission may be used by advanced clients.


A content author may signal such seamless switching property across [adaptation sets](#) by providing a [supplemental property descriptor](#) along with an [adaptation set](#) with `@schemeIdUri` set to `urn:mpeg:dash:adaptation-set-switching:2016` and the `@value` is a comma-separated list of [adaptation set](#) IDs that may be seamlessly switched to from this [adaptation set](#).

If the content author signals the ability of [adaptation set](#) switching and as `@segmentAlignment` or `@subsegmentAlignment` are set to true for one [adaptation set](#), the (sub)segment alignment shall hold for all [representations](#) in all [adaptation sets](#) for which the `@id` value is included in the `@value` attribute of the [supplemental property descriptor](#).

As an example, a content author may signal that seamless switching across an H.264/AVC [adaptation set](#) with `AdaptationSet@id="264"` and an HEVC [adaptation set](#) with `AdaptationSet@id="265"` is possible by adding a [supplemental property descriptor](#) to the H.264/AVC [adaptation set](#) with `@schemeIdUri` set to `urn:mpeg:dash:adaptationset-switching:2016` and the `@value="265"` and by adding a [supplemental property descriptor](#) to the HEVC [adaptation set](#) with `@schemeIdUri` set to `urn:mpeg:dash:adaptationset-switching:2016` and the `@value="264"`.

In addition, if the content author signals the ability of [adaptation set](#) switching for:

- any [video adaptation set](#) TODO
- any [audio adaptation set](#) TODO

ISSUE 18  What is the above talking about?

Note: This constraint may result that the switching may only be signaled with one [adaptation set](#), but not with both as for example one [adaptation set](#) signaling may include all spatial resolutions of another one, whereas it is not the case the other way round.

6.6. XLink[§]

Some XML elements in an MPD may be external to the MPD itself, delay-loaded by clients based on different triggers. This mechanism is called **XLink** and it enables client-side MPD composition from different sources. For the purposes of timing and addressing, it is important to ensure that the duration of each [period](#) can be accurately determined both before and after XLink resolution.

Note: XLink functionality in DASH is defined by [\[MPEGDASH\]](#) and [\[XLINK\]](#). This document provides a high level summary of the behavior and defines interoperability requirements.

XLink elements are those in the MPD that carry the `xlink:href` attribute. When XLink resolution is triggered, the client will query the URL referenced by this attribute. What happens next depends on the result of this query:

↔ **Non-empty result containing a valid XML fragment**

The entire [XLink element](#) is replaced with the query result. A single XLink element MAY be replaced with multiple elements of the same type.

↔ Empty result or query failure

The [XLink element](#) remains as-is with the XLink attributes removed.

When XLink resolution is triggered depends on the value of the `xlink:actuate` attribute. A value of `onLoad` indicates resolution at MPD load-time, whereas a value of `onRequest` indicates resolution on-demand at the time the client wishes to use the element. The default value is `onRequest`.

Services SHALL publish MPDs that conform to the requirements in this document even before XLink resolution. This is necessary because the behavior in case of XLink resolution failure is to retain the element as-is.

EXAMPLE 10

The below MPD example contains an XLink period. The real duration of the XLink [period](#) will only become known once the XLink is resolved by the client and the XLink element replaced with real content.

The first [period](#) has an explicit duration defined because the XLink resolver has no knowledge of the MPD and is unlikely to know the appropriate value to define for the second period's `Period@start` (unless this data is provided in the XLink URL as a parameter).

The explicitly defined duration of the second [period](#) will only be used as a fallback if the XLink resolver decides not to define a period. In this case the existing element in the MPD is preserved.

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:xlink="http://www.w3.org/1999/xlink" type="static">
  <Period duration="PT30S">
    ...
  </Period>
  <Period duration="PT0S" xlink:href="https://example.com/256479/clips/53473/as_period">
  </Period>
</MPD>
```

After XLink resolving, the entire `<Period>` element will be replaced, except when the XLink result is empty, in which case the client preserves the existing element (which in this case is a [period](#) with zero duration, ignored by clients).

Parts of the MPD structure that are not relevant for this chapter have been omitted - this is not a fully functional MPD file.

6.7. Update signaling via in-band events§

Services MAY signal the [MPD](#) validity duration by embedding in-band messages into [representations](#) instead of specifying a fixed validity duration in the MPD. This allows services to trigger [MPD refreshes](#) at exactly the desired time and to avoid needless [MPD refreshes](#).

The rest of this chapter only applies to services and clients that use in-band MPD validity signaling.

Services SHALL define `MPD@minimumUpdatePeriod=0` and add an in-band event stream to every audio [representation](#) or, if no audio [representations](#) are present, to every video [representation](#). The in-band event stream MAY also be added to other [representations](#). The in-band event stream SHALL be identical in every [representation](#) where it is present.

The in-band event stream SHALL be signaled on the [adaptation set](#) level by an `InbandEventStream` element with `@scheme_id_uri="urn:mpeg:dash:event:2012"` and a `@value` of 1 or 3, where:

- A value of 1 indicates that in-band events only extend the MPD validity duration.
- A value of 3 indicates that in-band events also contain the updated MPD snapshot when updates occur.

Services SHALL update `MPD@publishTime` to a unique value after every MPD update.

Note: `MPD@publishTime` is merely a version label. The value is not used in timing calculations.

EXAMPLE 11

Using in-band signaling and `MPD@minimumUpdatePeriod=0`, each [media segment](#) increases the validity period of the [MPD](#) by the duration of the [media segment](#) by default. When a validity event arrives, it carries the validity end timestamp of the [MPD](#), enabling the client to determine when a new [MPD refresh](#) is needed.

For a detailed definition of the mechanism and the event message data structures, see [\[MPEGDASH\]](#). This chapter is merely a high level summary of the most important aspects relevant to interoperability.

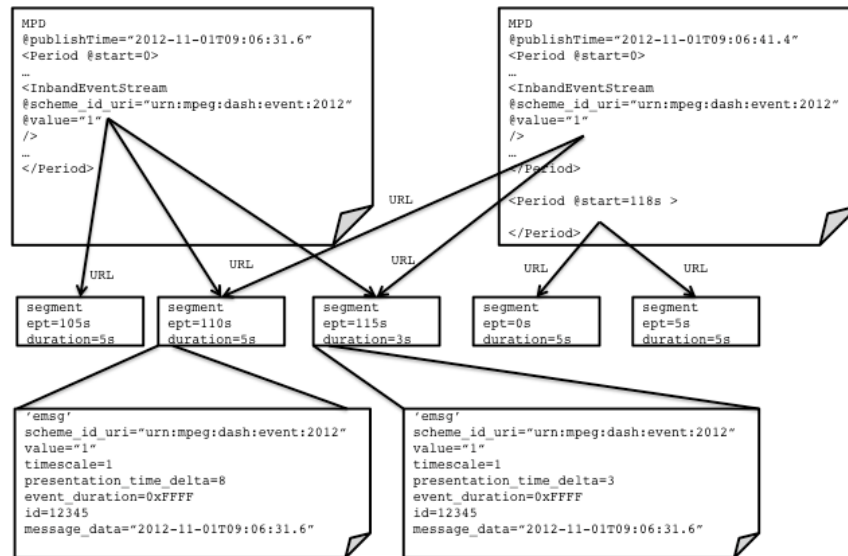


Figure 31 Illustration of MPD expiration signaling using in-band events.

Services SHALL emit in-band events as [\[MPEGDASH\]](#) emsg boxes to signal the [MPD](#) validity duration using the following logic:

- Lack of an in-band MPD validity event in a [media segment](#) indicates that an MPD that was valid at the start of the [media segment](#) remains valid up to the end of the [media segment](#).
- The presence of an in-band MPD validity event in a [media segment](#) indicates that the MPD with `MPD@publishTime` equal to the event's `publish_time` field remains valid up to the event start time.

The in-band events used for signaling MPD validity duration SHALL have `scheme_id_uri` and `value` matching the `InbandEventStream` element. Clients SHALL NOT use in-band events for MPD validity update signaling if these fields on the events do not match the `InbandEventStream` element or if the `InbandEventStream` element is not present in the [MPD](#).

In-band events with `value=3` SHALL provide an updated MPD in the event's `mpd` field as UTF-8 encoded text without a byte order mark.

Clients MAY perform [MPD refreshes](#) or process an event-embedded [MPD](#) immediately upon reading the event, without waiting for the moment signaled by the event timestamp. Services SHALL ensure that an updated [MPD](#) is available and valid starting from the moment a validity event is signaled.

Multiple [media segments](#) MAY signal the same validity update event (identified by matching `id` field on event), enabling the signal to be delivered several segments in advance of the MPD expiration.

In-band MPD validity events SHALL NOT be signaled in a static MPD but MAY be present in the [media segments](#) referenced by a static MPD, in which case they SHALL be ignored by clients.

Note: The above may happen when a live service is converted to an on-demand service for catchup/recording purposes.

6.8. Specifying initial position in presentation URL§

ISSUE 19 This section could use another pass to make it easier to read.

By default, a client would want to start playback from the start of the presentation (if `MPD@type="static"`) or from near the live edge (if `MPD@type="dynamic"`). However, in some situations it may be desirable to instruct clients to start

playback from a specific position. In [live services](#), where content has a fixed mapping to real time, this means an initial time-shift is applied.

The interoperable mechanism for this is to add an MPD anchor to the presentation URL. Details of this feature are defined in [\[MPEGDASH\]](#), with this chapter offering a summary of the feature and constraining its use to interoperable cases.

An initial position MAY be signalled to the DASH client by including an MPD anchor in the presentation URL. If an anchor is used, it SHALL be specified with one of the following sets of parameters:

- the `t` parameter
- both the `period` and `t` parameter

The `t` parameter indicates offset from [period](#) start or a moment in real-time, with `period` referencing a `Period@id` (defaulting to the first period).

The value of `Period@id` must be URL-encoded.

The time indicated using the `t` parameter SHALL be a single `npttime` value as specified in [\[media-frags\]](#). This is a narrower definition than accepted by [\[MPEGDASH\]](#).

EXAMPLE 12

To start from the beginning of the first [period](#) the following would be added to the end of the MPD URL provided to the DASH client: `#t=0`

To start with a fixed offset from the start of a specific [period](#), in this case 50 minutes from the beginning of the period with ID `program_part_2`, use the following syntax: `#period=program_part_2&t=50:00`

When accessing a [live service](#), you can instruct the client to use an initial time-shift so that content from a specific moment is played back by providing a POSIX timestamp with the `t` parameter. For example, starting playback from Wed, 08 Jun 2016 17:29:06 GMT would be expressed as `#t=posix:1465406946`. Starting playback from the live edge can be signalled as `#t=posix:now`.

When referencing a moment in real time using `t=posix`, the `period` parameter SHALL NOT be used.

ISSUE 20 How do leap seconds tie into this? See #161

7. Content annotation and selection

[\[MPEGDASH\]](#) enables a service to annotate [adaptation sets](#) to enable clients to make an informed decision on which [adaptation set](#) to select for presentation from among the alternatives offered for each [adaptation set type](#). The selection is based on client capabilities, client preferences, user preferences and possibly also interactive choices presented to the user. Typically, the signalling and selection is independent of the codec in use.

This chapter defines requirements and recommendations for annotating [adaptation sets](#) with interoperable descriptive information.

A service may offer multiple [adaptation sets](#) of the same type to provide the same content in different encodings or different source formats (e.g. one [adaptation set](#) encoded from a standard dynamic range master and another encoded from a high dynamic range video master). Alternatively, [adaptation sets](#) may describe different content (e.g. different languages or different camera views).

Note: While the typical situation is that a client selects one [adaptation set](#) per [adaptation set type](#), there may be cases where multiple [adaptation sets](#) of the same type are chosen for playback (e.g. [§ 6.5 Switching across adaptation sets](#)).

Proper annotation of [adaptation sets](#) in [MPDs](#) is essential in order to enable interoperable client implementations.

7.1. Annotations for content selection

[\[MPEGDASH\]](#) provides many options for annotating [adaptation sets](#). This document defines a restricted subset considered interoperable by DASH-IF members.

The table below lists the permitted annotations for each [adaptation set type](#). It is expected that interoperable DASH clients recognize the descriptors, elements, and attributes as documented in this chapter.

Content selection annotations SHALL be defined by a service in sufficient detail to differentiate every [adaptation set](#) from others of the same type. A service SHALL limit content selection annotations to those defined in this chapter.

Many of these annotations are defined by [\[MPEGDASH\]](#). Other organizations may define additional descriptors or elements. Some are defined by [JOP](#).

Note: [Supplemental property descriptors](#) are intended for presentation optimization and are intentionally not listed as annotations to be used for content selection.

Attribute or element	Use	Usage requirements
@profiles	O	If not present, it is inherited from the MPD or period . This may be used for example to signal extensions for new media profiles in the MPD.
@group	OD default=unique (see [MPEGDASH])	The attribute MAY be used. If present, it SHALL be greater than 0. The value SHALL be different for different adaptation set types and MAY be different for adaptation sets of the same type. This attribute enables a service to define logical groupings of adaptation sets . A client SHALL select either zero or one adaptation sets from each group.
@selectionPriority	OD default=1	This attribute SHOULD be used to express the preferences of the service on selecting adaptation sets for which the DASH client does make a decision otherwise. Examples include two video codecs providing the same content, but one of the two provides higher compression efficiency and is therefore preferred.
ContentProtection	0...N	If this element is present, then the content is protected. If not present, no content protection is applied. See § 12 Content protection and security
EssentialProperty	0...N	Defines an annotation that is considered essential for processing the adaptation set . See also essential property descriptor . Clients SHALL NOT select adaptation sets that are annotated with any instances of this element that are not understood by the client. The following schemes are expected to be recognized by a client independent of the adaptation set type: <ul style="list-style-type: none"> • http://dashif.org/guidelines/trickmode (see § 6.3 Trick mode)
Viewpoint	0...N	Indicates that adaptation set differentiates by a different viewpoint or combination of viewpoints. If present then all adaptation sets of the same type SHALL carry this descriptor with the same @schemeIdUri and different @value .
Label	0...N	Provides a textual description of the content. This element SHOULD be used if content author expects a client to support a UI for selection. If present then all adaptation sets of the same type SHALL carry this element with different values. This element SHALL NOT be used as the sole differentiating element, as scenarios with no user interaction must still lead to unambiguous selection.

Figure 32 Content selection annotations for any [adaptation set](#) type.

The following annotations are specific to an [adaptation set](#) type.

Attribute or element	Use	Usage requirements specific to video adaptation sets
@codecs	1...N	<p>Defines the codec that is necessary to present one or more representations in an adaptation set.</p> <p>This attribute can be present on either the adaptation set level (as a single value) or the representation level (in which case multiple values might be present).</p> <p>See § 11 Media coding technologies for a description of interoperable codecs.</p>
@par	M	The display aspect ratio at which content is intended to be displayed.
@maxWidth	O	<p>This attribute should be present to express the maximum width in samples after decoder sample cropping of any representation contained in the adaptation set.</p> <p>The value should be the maximum horizontal sample count of any SPS in the contained bitstream.</p>
@maxHeight	O	<p>This attribute should be present to express the maximum height in pixel of any representation contained in the adaptation set.</p> <p>The value should be the maximum horizontal sample count of any SPS in the contained bitstream.</p>
@maxFrameRate	O	This attribute should be present to express the maximum frame rate, i.e. the maximum value of any entry in the decoder configuration record of the signaled frame rate, if constant frame rate is provided. contained in the adaptation set .
EssentialProperty	0...N	<p>Defines an annotation that is considered essential for processing the adaptation set. See also essential property descriptor.</p> <p>Clients SHALL NOT select adaptation sets that are annotated with any instances of this element that are not understood by the client.</p> <p>The following schemes are expected to be recognized by a client for video adaptation sets:</p> <ul style="list-style-type: none"> urn:mpeg:mpegB:cicp:<Parameter> as defined in [iso23001-8] and <Parameter> being one of the following: ColourPrimaries, TransferCharacteristics, Or MatrixCoefficients
Accessibility	0...N	<p>Defines the type of accessibility-relevant content present in the adaptation set.</p> <p>The set of descriptors SHALL be restricted to the following:</p> <ul style="list-style-type: none"> The "Role" scheme as defined by [MPEGDASH], with @schemeIdUri="urn:mpeg:dash:role:2011". A client is expected to recognize the following values when this scheme is used in the Accessibility descriptor: <ul style="list-style-type: none"> sign caption The CEA-608 scheme with @schemeIdUri="urn:scte:dash:cc:cea-608:2015" (see § 11.8 CEA-608/708 Digital Television (DTV) Closed Captioning)
		Defines the role of the content in the adaptation set .

Attribute or element	Use	Usage requirements specific to video adaptation sets
Role	0...N	<p>Usage requirements SHALL be restricted to the "Role" scheme as defined by [MPEGDASH] with @schemeIdUri="urn:mpeg:dash:role:2011" MAY be used for differentiation. A client is expected to recognize the following values when this scheme is used in the Role descriptor:</p> <ul style="list-style-type: none"> caption subtitle main alternate supplementary sign emergency <p>Clients SHALL consider there to be an implicit Role descriptor with the "Role" scheme and the value main if no explicitly defined Role descriptor with the "Role" scheme is present.</p>

Figure 33 Annotations for [video adaptation sets](#).

Attribute or element	Use	Usage requirements specific to audio adaptation sets
@codecs	1...N	<p>Defines the codec that is necessary to present one or more representations in an adaptation set.</p> <p>This attribute can be present on either the adaptation set level (as a single value) or the representation level (in which case multiple values might be present).</p> <p>See § 11 Media coding technologies for a description of interoperable codecs.</p>
@lang	M	The language of the audio stream.
@audioSamplingRate	M	The audio sampling rate.
AudioChannelConfiguration	1...N	<p>specifies information about the audio channel configuration. The following schemes are expected to be recognized by a client:</p> <ul style="list-style-type: none"> urn:mpeg:dash:23003:3:audio_channel_configuration:2011 as defined in [MPEGDASH]. urn:mpeg:mpegB:cicp:ChannelConfiguration as defined in [iso23001-8]. tag:dolby.com,2014:dash:audio_channel_configuration:2011 as defined in the DASH-IF identifier registry
EssentialProperty	0...N	<p>Defines an annotation that is considered essential for processing the adaptation set. See also essential property descriptor.</p> <p>Clients SHALL NOT select adaptation sets that are annotated with any instances of this element that are not understood by the client.</p> <p>The following schemes are expected to be recognized by a client for audio adaptation sets:</p> <ul style="list-style-type: none"> urn:mpeg:dash:audio-receiver-mix:2014 as defined in [MPEGDASH]
		<p>Defines the type of accessibility-relevant content present in the adaptation set.</p> <p>The set of descriptors SHALL be restricted to the "Role" scheme as defined by [MPEGDASH], with</p>

Accessibility Attribute or element	0..N Use	@schemeIdUri="urn:mpeg:dash:role:2011". A client is expected to recognize the following values when this scheme is used in the Usage requirements specific to audio adaptation sets
		<p>Accessibility descriptor:</p> <ul style="list-style-type: none"> • description • enhanced-audio-intelligibility
Role	0..N	<p>The set of descriptors SHALL be restricted to the "Role" scheme as defined by [MPEGDASH] with @schemeIdUri="urn:mpeg:dash:role:2011" MAY be used for differentiation. A client is expected to recognize the following values when this scheme is used in the Role descriptor:</p> <ul style="list-style-type: none"> • main • alternate • supplementary • commentary • dub • emergency <p>Clients SHALL consider there to be an implicit Role descriptor with the "Role" scheme and the value main if no explicitly defined Role descriptor with the "Role" scheme is present.</p>

Figure 34 Annotations for [audio adaptation sets](#).

Attribute or element	Use	Usage requirements specific to text adaptation sets
@codecs	0..N	<p>Defines the codec that is necessary to present one or more representations in an adaptation set.</p> <p>This attribute can be present on either the adaptation set level (as a single value) or the representation level (in which case multiple values might be present).</p> <p>The attribute SHALL be present, except when IOP does not define a <code>@codecs</code> value for the used text codec and encapsulation mode combination, in which case it SHALL be omitted.</p> <p>See § 11 Media coding technologies for a description of interoperable codecs.</p>
@lang	M	The text language.
Accessibility	0..N	<p>Defines the type of accessibility-relevant content present in the adaptation set.</p> <p>The set of descriptors SHALL be restricted to the "Role" scheme as defined by [MPEGDASH], with <code>@schemeIdUri="urn:mpeg:dash:role:2011"</code>. A client is expected to recognize the following values when this scheme is used in the Accessibility descriptor:</p> <ul style="list-style-type: none"> • sign • caption
Role	0..N	<p>Defines the role of the content in the adaptation set.</p> <p>The set of descriptors SHALL be restricted to the "Role" scheme as defined by [MPEGDASH] with <code>@schemeIdUri="urn:mpeg:dash:role:2011"</code> MAY be used for differentiation. A client is expected to recognize the following values when this scheme is used in the Role descriptor:</p> <ul style="list-style-type: none"> • main • alternate • subtitle • supplementary • commentary • dub • description • emergency <p>Clients SHALL consider there to be an implicit Role descriptor with the "Role" scheme and the value <code>main</code> if no explicitly defined Role descriptor with the "Role" scheme is present.</p>

Figure 35 Annotations for [text adaptation sets](#).

7.2. Content models

In order to support the content author in providing content in a consistent manner, this chapter provides a conceptual content model for DASH content in one [period](#) of an [MPD](#). The content may be described by an [asset identifier](#) as a whole and may contain different [adaptation set types](#).

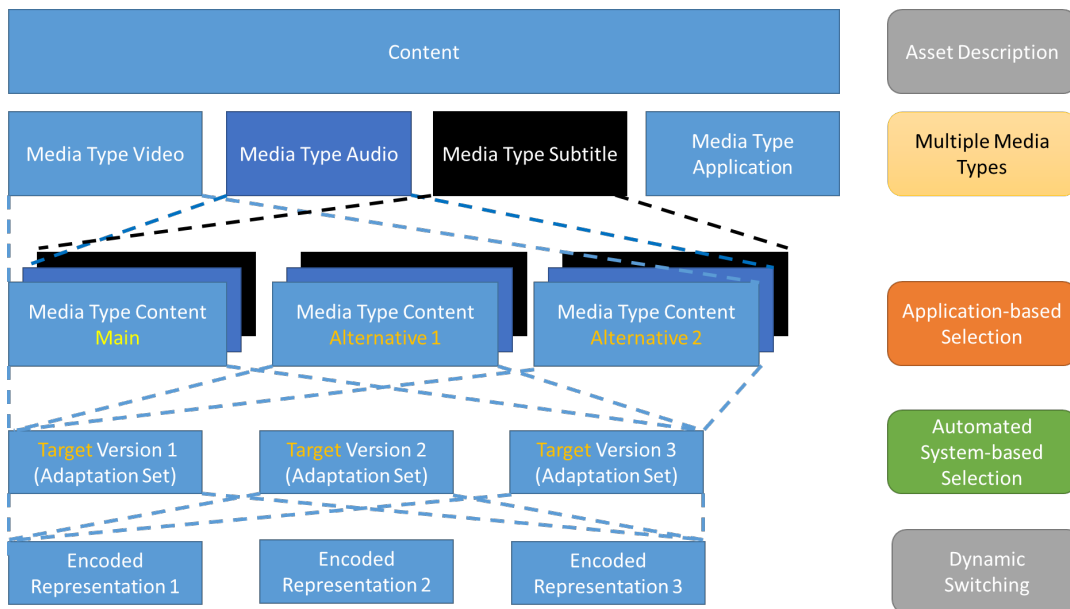


Figure 36 Model for content selection.

Within each [adaptation set](#) type, the content author may want to offer alternative content that is time-aligned but where each alternative represents different content (e.g. multiple camera angles). Automatic selection of the alternative content is not expected to be done by the DASH client as the client would not have sufficient information to make such decisions. However, the selection is expected to be done by communication with an application or the user, typically using a user interface appropriate for selection.

In the absence of user indication to select from among the alternatives, the DASH client still needs to select content to be presented. A DASH service must therefore signal the preferred default content. The preferred content is referred to as **main content**, whereas any content that is not [main content](#) is referred to as **alternative content**. There may be multiple alternatives which may need to be distinguished. See [§ 7.2.1 Signaling alternative content](#).

Furthermore, it may be that content of different [adaptation set types](#) is linked by the content author, to express that two content of different [adaptation set](#) type are preferably played together. We define **associated content** for this purpose. As an example, there may be a main commentator associated with the main camera view, but for a different camera view, a different associated commentary is provided. See [§ 7.2.2 Signaling associated content](#).

In addition to semantical content level differentiation, each [alternative content](#) may be provided in different variants, based on content preparation properties (downmix, subsampling, translation, suitable for trick mode, etc.), client preferences (decoding or rendering preferences, e.g. codec), client capabilities (DASH profile support, decoding capabilities, rendering capabilities) or user preferences (accessibility, language, etc.). Both [main content](#) and [alternative content](#) in all their variants are differentiated in the [MPD](#) as defined in [§ 7.1 Annotations for content selection](#).

7.2.1. Signaling alternative content

If a [period](#) contains alternative content for one [adaptation set](#) type, then the alternatives SHALL be differentiated according to [§ 7.1 Annotations for content selection](#) and one of the alternatives SHALL be provided as [main content](#).

[Main content](#) is signaled by using the `role` descriptor with scheme `urn:mpeg:dash:role:2011` and value set to `main`. Alternative content is signaled by using the `role` descriptor with scheme `urn:mpeg:dash:role:2011` and value set to `alternative`.

7.2.2. Signaling associated content

A [Viewpoint](#) descriptor with the same `@schemeIdUri` and `@value` SHALL be used by services to signal [associated content](#).

Clients SHALL use identical [Viewpoint](#) descriptors for determining associated content even if they do not understand the `@schemeIdUri`.

7.3. Client processing reference model

The following client model serves two purposes:

- In the absence of other information, the following client model may be implemented in a DASH client for the purpose of selection of [adaptation set](#) for playout
- A content author may use the model to verify that the annotation is properly done in order to get the desired client behaviour.

In the model it is assumed that the client can get sufficient information on at least the following properties:

- For each codec in the `@codecs` string, the DASH client can get information if the media playback platform can decode the codec as described in the string. The answer should be yes or no.
- For each DRM system in the `ContentProtection` element string, the DASH client can get information if the media playback platform can handle this content protection scheme as described in the string. The answer should be yes or no.
- the DASH client can get information on the media playback platform and rendering capabilities in terms of of
 - the maximum spatial resolution for video that can be handled
 - the maximum frame rate for video that can be handled
 - the audio channel configuration of the audio system
 - the audio sampling rate of the audio system
- the preferred language of the system
- Accessibility settings for captions, subtitles, audio description, enhanced audio intelligibility,
- Potentially preferences on media playback and rendering of the platform

Note: If any of these functionalities are not fulfilled, then the client may still be functional, but it may not result in the full experience as provided by the content author. As an example, if the DASH client cannot determine the preferred language, it may just use the selection priority for language selection.

The DASH client uses the MPD and finds the [period](#) that it likes to join, typically the first one for On-Demand content and the one at the live edge for live content. In order to select the media to be played, the DASH client assumes that the content is offered according to the content model above.

1. The DASH client looks for [main content](#), i.e. any [adaptation set](#) with annotation `Role@schemeIdUri="urn:mpeg:dash:role:2011"` and `Role@value="alternative"` is excluded initially for selection. Note that in this model it is assumed that immediate startup is desired. If the DASH client wants to go over the alternatives upfront before starting the service, then the sequence is slightly different, but still follows the remaining principles.
2. DASH Client checks each [adaptation set](#) for the supported capabilities of the platform. If any of the capabilities are not supported, then the [adaptation set](#) is excluded from the selection process.
 - Codec support
 - DRM support
 - Rendering capabilities
3. The DASH client checks if it supports for CEA-608 rendering as defined in clause [§ 11.8 CEA-608/708 Digital Television \(DTV\) Closed Captioning](#). If not supported, any accessibility descriptor with `@schemeIdUri="urn:scte:dash:cc:cea-608:2015"` is removed. Note that the [adaptation set](#) is maintained as it may be used for regular video decoding.
4. DASH Client checks if there are any specific settings for accessibility in the user preferences
 - If captions are requested by the system, the DASH client extracts
 - all video [adaptation sets](#) that have an `Accessibility` descriptor assigned with either the `@schemeIdUri="urn:mpeg:dash:role:2011"` and `@value="caption"` or `@schemeIdUri="urn:scte:dash:cc:cea-608:2015"` (burned-in captions and SEI-based), as well as
 - all [text adaptation sets](#) that have an `Accessibility` descriptor assigned with either the `@schemeIdUri="urn:mpeg:dash:role:2011"` and `@value="caption"`
 - and makes those available for [adaptation sets](#) that can be selected by the DASH client for caption support.
 - If multiple [text adaptation sets](#) remain, the DASH client removes all [adaptation sets](#) from the selection that are not in the preferred language, if language settings are provided in the system. If no language settings in the system are provided, or none of the [adaptation sets](#) meets the preferred languages,

none of the [adaptation sets](#) are removed from the selection. Any [adaptation sets](#) that do not contain language annotation are removed, if any of the remaining [adaptation sets](#) provides proper language settings.

- If still multiple [text adaptation sets](#) remain, then the ones with the highest value of `@selectionPriority` are chosen.
 - If still multiple [text adaptation sets](#) remain, then the DASH client makes a random choice on which caption to enable.
 - else if no captions are requested
 - the `Accessibility` element signaling captions may be removed from the [adaptation set](#) before continuing the selection.
 - If sign language is requested
 - all [video adaptation sets](#) that have an `Accessibility` descriptor assigned with `@schemeIdUri="urn:mpeg:dash:role:2011"` and `@value="sign"` are made available for sign language support.
 - else if no sign language is requested
 - the [adaptation set](#) signaling sign language with the `Accessibility` element may be removed from the [adaptation set](#) before continuing the selection
 - If audio descriptions are requested
 - all [video adaptation sets](#) that have an `Accessibility` descriptor assigned with `@schemeIdUri="urn:mpeg:dash:role:2011"` and `@value="description"` are made available for audio description support.
 - else if no audio descriptions are requested
 - the [adaptation set](#) signaling audio descriptions with the `Accessibility` element may be removed from the [adaptation set](#) before continuing the selection.
 - If enhanced audio intelligibility is requested
 - all [audio adaptation sets](#) that have an `Accessibility` descriptor assigned with `@schemeIdUri="urn:mpeg:dash:role:2011"` and `@value="enhanced-audio-intelligibility"` are made available for enhanced audio intelligibility support.
 - else if no enhanced audio intelligibility is requested
 - the `Accessibility` element may be removed from the [adaptation set](#) before continuing the selection.
5. If video rendering is enabled, based on the remaining [video adaptation sets](#) the client selects one as follows:
- Any [adaptation set](#) for which an [essential property descriptor](#) is present for which the scheme or value is not understood by the DASH client, is excluded from the selection
 - Any [adaptation set](#) for which an [essential property descriptor](#) is present for which the scheme is `http://dashif.org/guidelines/trickmode`, is excluded from the initial selection
 - If still multiple [video adaptation sets](#) remain, then the ones with the highest value of `@selectionPriority` is chosen.
 - If still multiple [video adaptation sets](#) remain, then the DASH client makes a choice for itself, possibly on a random basis.
 - Note that an [adaptation set](#) selection may include multiple [adaptation sets](#), if [adaptation set switching](#) is signaled. However, the selection is done for only one [adaptation set](#).
6. If audio rendering is enabled, based on the remaining [audio adaptation sets](#) the client selects one as follows:
- Any [adaptation set](#) for which an [essential property descriptor](#) is present for which the scheme or value is not understood by the DASH client, is excluded from the selection
 - If multiple [audio adaptation sets](#) remain, the DASH client removes all [adaptation sets](#) from the selection that are not in the preferred language, if language settings are provided in the system. If no language settings in the system are provided, or none of the [adaptation sets](#) meets the preferred languages, none of the [adaptation sets](#) are removed from the selection. Any [adaptation set](#) that does not contain language annotation are removed, if any of the remaining [adaptation sets](#) provides proper language settings
 - If still multiple [audio adaptation sets](#) remain, then the ones with the highest value of `@selectionPriority` are chosen
 - If still multiple [audio adaptation sets](#) remain, then the DASH client makes a choice for itself, possibly on a random basis

- Note that an [adaptation set](#) may include multiple [adaptation sets](#), if [adaptation set switching](#) or receiver mix is signaled. However, the selection is done for only one [adaptation set](#).
7. If text rendering is enabled, based on the [text adaptation sets](#) the client selects one as follows:
 - Any [adaptation set](#) for which an [essential property descriptor](#) is present for which the scheme or value is not understood by the DASH client, is excluded from the selection
 - If multiple [text adaptation sets](#) remain, the DASH client removes all [adaptation sets](#) from the selection that are not in the preferred language, if language settings are provided in the system. If no language settings in the system are provided, or none of the [adaptation sets](#) meets the preferred languages, none of the [adaptation sets](#) are removed from the selection. Any [adaptation set](#) that does not contain language annotation are removed, if any of the remaining [adaptation sets](#) provides proper language settings.
 - If still multiple [text adaptation sets](#) remain, then the ones with the highest value of `@selectionPriority` are chosen.
 - If still multiple [text adaptation sets](#) remain, then the DASH client makes a choice for itself, possibly on a random basis.
 8. If the DASH client has the ability to possibly switch to alternative content, then alternative content may be selected either through the `Label` function or the `Viewpoint` functionality. This selection may be done dynamically during playout and the DASH client is expected to switch to the alternative content. Once all alternative content is selected, the procedures following from step 2 onwards apply.
 9. At [period](#) boundary a DASH client initially looks for [period continuity or connectivity](#), i.e. does the [period](#) include an [adaptation set](#) that is a continuation of the existing one. If not present it will go back to step 1 and execute the decision logic.

8. On-demand services§

An on-demand service is one that is published with a static [MPD](#) (`MPD@type="static"`).

On-demand services have an infinite [availability window](#) and [have no fixed mapping to real time](#) - clients may present any part at any time and may use [trick mode](#) support to alter the playback rate.

Note: An on-demand service may be created by transforming what was previously a [live service](#) into an on-demand service for viewing as a catch-up presentation or a recording. See [§ 9.10 Converting a live service to an on-demand service](#).

On-demand services MAY use any [addressing mode](#) or even a combination of multiple [addressing modes](#).


[MPD](#) elements or attributes only relevant for dynamic [MPDs](#) SHALL NOT be present in [MPDs](#) of on-demand services. Clients SHALL ignore any such elements or attributes if present.

8.1. Surviving transforming boxes and other adaptation middleboxes§

A number of video transcoding proxies (aka "middleboxes") are deployed on the Internet that may silently transcode DASH presentations. Specifically, a middlebox may see a `video/mp4` HTTP response, transcode that video into a different format (perhaps using a lower bitrate or a different codec), then forward the transcoded video to the DASH client. This will break byte range based operations, as byte ranges from the original video are not valid in the transcoded video.

If such a threat is encountered, the following options may prevent proxies from transcoding DASH presentations:

- Serve DASH presentations using an authenticated transport that prevents interception (HTTPS).
- Serve DASH presentations using encryption that prevents tampering.
- Serve DASH presentations with the `Cache-Control: no-transform` header.

ISSUE 22  Insert reference to encryption.

In all cases the operational impacts on caching and implementations should be considered when using any of the above technologies. The same methods may also need to be applied to prevent middleboxes manipulating the [MPD](#).

9. Live services§

A live service is one that is published with a dynamic [MPD](#) (`MPD@type="dynamic"`).

Live services have a [strict mapping between the MPD timeline and real time](#) and are often [available](#) only for a limited time. The [MPD](#) of a live service [may change over time](#), for example as more content is appended and expired content is removed. Clients are forced into a timed schedule for the playout, such that they follow the schedule as desired by the content author (with some amount of [client-controlled time-shift](#) allowed).

A live service has a **live edge**, which is the most recent moment on the [MPD timeline](#) for which the [MPD](#) guarantees that [media segments](#) are [available](#) for all [representations](#). See [§ 9.7 Determining the live edge](#).

Live services MAY use either [explicit addressing](#) or [simple addressing](#) or a combination of the two. [Indexed addressing](#) is not meaningful in a live service context.

Note: In previous versions of [IOP](#) a distinction was made between "simple live" and "main live" services. The latter simply refers to live services that signal [MPD updates using in-band events](#).

There are multiple different types of live services:

Scheduled playback of prepared content

The content is prepared in advance but playback is scheduled for a specific time span in real time.

MPD-controlled live service

The content is generated on the fly and the [MPD](#) receives constant updates to reflect the latest state of the service offering. The DASH client behavior is driven solely by the [MPD](#) contents, which it regularly refreshes.

MPD- and segment-controlled live service

The content is generated on the fly and clients are kept informed of [MPD](#) validity by [in-band events](#) in the [media segments](#). [MPD](#) downloads are only initiated when the need for updates is detected. Services can signal the need for updates on short notice.

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. An [MPD anchor](#) MAY be used when referencing the [MPD](#) to specify an initial time-shift that clients are expected to apply.

Note: Support for [MPD anchors](#) is an optional client feature - a service should consider clients that lack an implementation.

The initial [MPD](#) or join [MPD](#) is accessed and processed by the client and, [having an accurate clock that is synchronized with the server](#), the client can analyze the [MPD](#) and extract suitable information in order to initiate playback of the service. This includes, but is not limited to:

- Identifying the currently active [periods](#) in the service and the [period](#) that contains the [live edge](#).
- 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 [representation](#). The client then issues requests for the [media 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 [media segments](#). See [§ 5.2.9.5 MPD updates](#) and [§ 6.7 Update signaling via in-band events](#).

9.1. Selecting the time shift buffer size§

Recommended configuration for [time shift buffer](#) size:

If playback should only occur near the live edge, without significant time shift possibility.

`MPD@timeShiftBufferDepth` SHOULD be short but with a lower limit of 4 times [media segment](#) duration or 6 seconds (whichever is largest). This gives the client some opportunity to time-shift for buffering purposes, to overcome difficult network conditions.

If playback is not limited to near-live-edge.

`MPD@timeShiftBufferDepth` MAY have an arbitrarily large value, including a value greater than the total duration of [periods](#) in the presentation.

9.2. Selecting the suggested presentation delay§

Recommended configuration for [presentation delay](#):

If the service wishes to explicitly synchronize playback of different clients.

MPD@suggestedPresentationDelay SHOULD be set to the desired [presentation delay](#) but with a lower limit of 4 seconds or 2-4 times the [media segment](#) duration (whichever is largest).

If the service does not wish to explicitly synchronize playback of different clients.

Omit MPD@suggestedPresentationDelay and let each client determine the optimal [presentation delay](#) based on its own heuristics (which may lead different clients to choosing a different [presentation delay](#)).

The limitations imposed by the following factors SHOULD be considered when selecting the value for the [presentation delay](#):

- The desired end-to-end latency.
- The typical amount of buffering that must be performed by clients.
- [The minimum buffer time](#).
- The [time shift buffer](#) size.

9.3. Selecting the media segment duration§

The [media segment](#) duration SHOULD be between 1 and 10 seconds. The duration influences the end-to-end latency but also the switching and random access granularity as in DASH-264/AVC each [media segment](#) starts with a stream access point which can also be used as a [switching 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.

9.4. Safety margins in availability timing§

There exists unavoidable jitter and occur occasional delays in most content delivery architectures. A DASH client SHOULD avoid being too aggressive in requesting [media segments](#) as soon as they become [available](#). If a DASH client observes issues, such as 404 responses, it SHOULD back up slightly in the requests.

Services SHALL be published so that all timing promises made by the [MPD](#) hold under normal operating conditions. Services MAY indicate an [availability window](#) that includes a safety margin. However, such a safety margin will lead to increased end-to-end latency, so it is a balance to be taken into account.

If a service wishes to impose a safety margin of N seconds, it SHOULD offset MPD@availabilityStartTime into the future by N seconds when starting the presentation.

9.5. Selecting the minimum update period§

The minimum update period signals that MPD@minimumUpdatePeriod worth of future [media segments](#) are guaranteed to become [available](#) over that time span after retrieving the [MPD](#).

Setting the value of the minimum update period primarily affects two main aspects of a service:

- 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.

The downside of a small minimum update period is that a large number of [MPD](#) download requests will be made by clients. This overhead can be minimized by [conditional GET requests](#) and/or [in-band MPD update signaling](#).

If [in-band MPD validity signaling](#) is used, MPD@minimumUpdatePeriod SHALL be 0.

9.6. Robust and seamless period transitions§

Multilanguage live services are likely to encounter experience transitions from one [period](#) to another. For example, due to changes in the set of available audio/text languages or due to ad insertion.

To ensure robust client operation at period transitions, ensure that all the requirements of the [timing model](#) are satisfied. In particular, periods must be fully covered by [media segment](#) references and media samples, including immediately before/after a period transition. No gaps can occur in any [representation](#)!

In many of these cases, some [adaptation sets](#) are likely to continue seamlessly across period boundaries, in which case they SHOULD be marked as [period-connected or period-continuous](#).

9.7. Determining the live edge§

If a service does not declare a suggested [presentation delay](#) or if the client chooses to ignore it, the client will likely want to know the position of the [live edge](#) in order to perform its own [presentation delay](#) calculations.

The [live edge](#) is affected by the following factors:

- The [time shift buffer](#) determines the set of [media segments](#) that a client may present.
- However, not all [media segments](#) may be available yet, even if they are within the [time shift buffer](#)!
- Furthermore, different [representations](#) may have different [availability windows](#) (due to `@availabilityTimeOffset`).
- And finally, different [representations](#) may have different [media segment](#) lengths (even within the same [representation](#)), which is important because [media segments](#) only become [available](#) once their **end point** is within the [availability window](#).

Accordingly, the [live edge](#) can be calculated as follows:

1. Determine the maximum [media segment](#) length `segment_length_max` for each [representation](#).
 - With [indexed addressing](#), the index segment provides the exact length of every [media segment](#).
 - With [simple addressing](#), the [MPD](#) defines a constant length but remember that [simple addressing allows for segment length deviation up to 200% of nominal length](#).
 - With [explicit addressing](#), every [media segment](#) may have a unique length and either `MDP@maxSegmentDuration` (if present) or custom heuristics should be used.
2. Determine the [availability window](#) end position `availability_end` for each representation.
3. Determine the minimum guaranteed start of the most recent [available media segment](#) `available_segment_start` for each representation as `available_segment_start = availability_end - segment_length_max`.
4. The [live edge](#) is `min(available_segment_start)`.

A client MAY exclude some [representations](#) from live edge calculation if it considers them optional for successful playback. For example, [trick mode representations](#) may become [available](#) in a delayed fashion and would needlessly delay the live edge. See also [§ 9.8 Trick mode for live services](#).

Note: When determining the [presentation delay](#) a client should also consider other aspects besides the [live edge](#) such as clock synchronization accuracy, expected network performance jitter and desired buffer size.

See also [§ 9.4 Safety margins in availability timing](#).

9.8. Trick mode for live services§

If trick mode is to be supported for [live services](#), the trick mode [representations](#) SHOULD be offered using the same [media segment](#) duration as in the main [adaptation set](#) or each [media segment](#) duration should aggregate an integer multiple of the [media segments](#) in the main [adaptation set](#).

The content author needs to find a balance between the [media segment](#) duration affecting the amount of requests in fast forward or fast rewind and the [availability](#) timing of trick mode [media segments](#). Longer [media segment](#) durations for the trick mode [representation](#) delay the [availability](#) time of such [media segments](#) by the duration of the [media segment](#) - i.e. at the [live edge](#) the trick mode may not be fully supported.

Based on this it is a content author's decision to provide one or more of the following alternatives for trick mode for live services:

- Provide one trick mode [adaptation set](#) that generates a [media segment](#) for every [media segment](#) in the main [adaptation set](#). Note that if this [adaptation set](#) is used, it may result in increased amount of HTTP requests when the player does a fast forward or fast rewind.

- Provide one trick mode [adaptation set](#) that generates a [media segment](#) only after several [media segments](#) in the main [adaptation set](#) have been generated and aggregate the trick mode samples in a single [media segment](#) of longer duration. As a result, it is possible that no trick mode samples are available at the [live edge](#) or that some clients consider the [live edge](#) to be where the trick mode [media segments](#) become [available](#).
- Provide multiple trick mode [adaptation sets](#) with different [media segment](#) durations, enabling trick mode aware clients to choose the desired tradeoff between efficiency and delay.
- Provide trick mode [adaptation sets](#) using [indexed addressing](#). This requires an entire [period](#) worth of data to be published simultaneously, so is only possible for [periods](#) that have a fixed end point and for which all data already exists.

Combinations of different alternatives are possible.

If a client wants to access a trick mode [adaptation set](#) in a live service, it SHOULD attempt to minimize the amount of requests to the network by preferring [media segments](#) with longer duration (if multiple choices are provided).

If a service is [converted from live to on-demand](#), trick mode [adaptation sets](#) SHOULD be converted to use [indexed addressing](#).

9.9. DVB-DASH alignment§

For alignment with [\[DVB-DASH\]](#), 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-BaseURL offering in section 10.8.2.1.

9.10. Converting a live service to an on-demand service§

The major difference between live and on-demand services is that live services have their timeline mapped to a real time clock and have an MPD that may change. This behavior is signaled by `MPD@type="dynamic"`. To transform a live service to an on-demand service, it may often be sufficient to set `MPD@type="static"` and to remove any signaling in the [MPD](#) that is restricted to dynamic MPDs.

There is no need to alter [media segments](#) when transforming a live service to an on-demand service.

Consider the time span of available content. A live service has a [time shift buffer](#) that may only allow a recent time span of content to be presented as a live service. If you wish to publish a larger time span as a recording, creating a separate on-demand variant of the [MPD](#) in parallel with the on-demand variant may be sensible.

A live service MAY be converted to an on-demand service without changing the URL, by simply replacing the dynamic MPD with a static MPD. Maintaining the same URLs for [media segments](#) might be beneficial in terms of CDN efficiency.

See also [§ 5.2.9.5.3 End of live content](#).

9.11. Reliable and consistent-delay live service§

ISSUE 23 This and everything below needs to be updated to conform to timing model

ISSUE 24 Needs proper Bikeshed formatting and references

A service provider wants to run a live DASH service according to the below Figure 8. As an example, a generic encoder for a 24/7 linear program or a scheduled live event provides a production encoded stream. Such streams typically include inband events to signal program changes, ad insertion opportunities and other program changes. An example for such signalling are SCTE-35 [54] messages. The stream is then provided to one or more Adaptive Bitrate (ABR) encoders, which transcodes the incoming stream into multiple bitrates and also conditions the stream for segmentation and program changes. These multiple encoders may be used for increased ABR stream density and/are then distributed downstream for redundancy purposes. The resultant streams are received by the DASH generation engines that include: MPD generator, packager and segmenter. Typically the following functions are applied by the MPD packager:

- Segmentation based on in-band information in the streams produced by the ABR encoders
- Encapsulation into ISO BMFF container to generate DASH segments
- Dynamic MPD generation with proper customization options downstream
- Event handling of messages
- Any other other DASH related adaptation

Downstream, the segments may be hosted on a single origin server, or in one or multiple CDNs. The MPD may even be further customized downstream, for example to address specific receivers. Customization may include the removal of certain Adaptation Sets that are not suitable for the capabilities of downstream clients. Specific content may be spliced based on regional services, targeted ad insertion, media blackouts or other information. Events carried from the main encoder may be interpreted and removed by the MPD packager, or they may be carried through for downstream usage. Events may also added as MPD events to the MPD.

In different stages of the encoding and distribution, errors may occur (as indicated by lightning symbols in the diagram), that for itself need to be handled by the MPD Generator and packager, the DASH client, or both of them. The key issue for this section is the ability for the DASH Media Presentation Generator as shown in to generate services that can handle the incoming streams and provide offerings such that DASH clients following DASH-IF IOPs can support.

Hence this section primarily serves to provide guidelines for implementation on MPD Generators and Packagers.

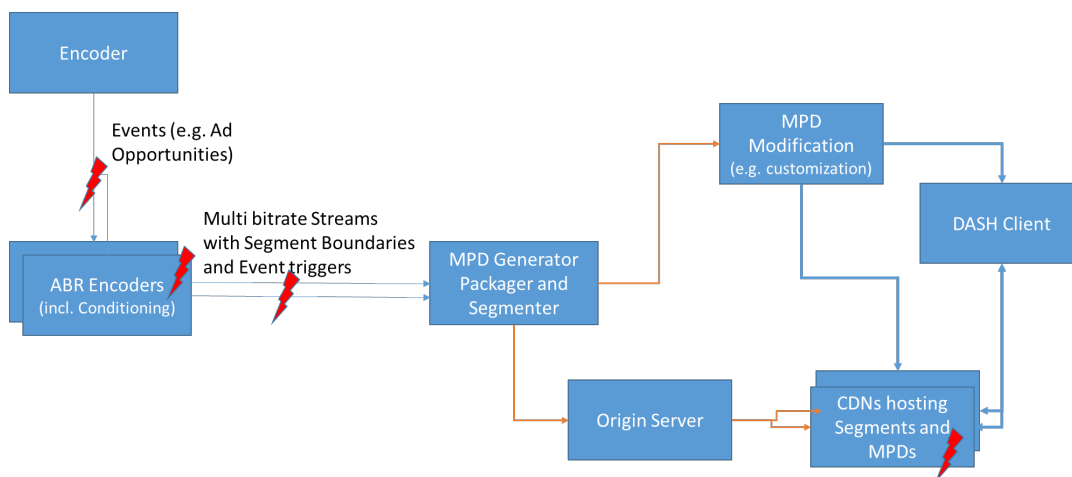


Figure 37 Example live service deployment architecture.

The following scenarios are considered in the service setup:

- The distribution latency should be consistent, typically what is observed for broadcast TV services. This means that the MPD Generator should add minimum delay, and the service should be setup such that the delay between MPD generator and DASH client playout is consistent, and preferably small.
- Program events may occur for different reasons, for example Program changes, switches from Programs to Advertisements or vice versa, media blackouts or other program changes. If [MPEG2TS](#) is used at broadcast origination points, the Program Map Table (PMT) typically indicates such changes. Typically, these changes also result in discontinuities for in the media timeline. Such changes are typically anticipated only on short notice, i.e. within a few seconds. In the following we refer to the time that changes are announced as change lead time. The service should also provide a minimum change lead time, i.e. the smallest time in media time between the change being announced in the stream and the time between the change occurs. Changes may for example include one or more of the following:
 - Number of source audio languages or formats can change. For example:
 - Programming with English and Spanish to other content with only English
 - Descriptive audio may disappear / reappear
 - Programming with 5.1 E-AC-3 and AAC Stereo content to other content with only Stereo AAC
 - Resolution or format of source video content can change, e.g. HD to/from SD, HDR to/from SDR, etc.
 - Codecs may change, or at least the profile or level of the codecs
 - The number of Representations in an Adaptation Set may change
 - A distribution network may be changed, added or removed.
- The segmentation is determined by the ABR encoders. This encoding may result in occasional slight variations

in segment durations during a period (as compared to the last segment in a period) due to encoding optimizations around scene changes near the segment duration point (for example: making a segment slightly shorter or longer to align segment IDR to a scene change).

- Unanticipated losses and operational failures or outages, possibly happen just for a single encoding (typically at the input of the encoder, but also possibly also downstream packaging).
 - Examples are
 - An encoder for one or more Representations or the output of an encoder fails for some time and does not produce content.
 - An encoder or the input to the encoder or the output of the encoder fails for a media component/Adaptation Set for some time and do not produce content.
 - All encoding or the input to the encoder fails for some time.
 - In all cases an MPD can still be written and the MPD is up and running. Also in the distribution, single Segments may be lost for different reasons and the client typically gets 404.
- MPD updates should be minimized, whereby MPD updates includes the following aspects for every MPD request
 - Client sending uplink requests for MPDs
 - Sending full MPD with every request from the server to the client
 - Parsing and processing of MPD at the client
 - Writing a new MPD on the server if the MPD is changed

ISSUE 25 Check and align references in original text.

The subchapters here outline some possibilities for solving the above challenges.

9.11.1. Consistent latency

The scenario does not ask for very low latency, but for consistent latency. Latency can primarily be controlled by the following means:

- Segment duration: the segment duration typically directly impacts the end-to-end latency. Smaller segment sizes provide improved latency and segments of 1-2 seconds may be chosen, if latency is an important aspect. However, too small segments may result in issues, as compression efficiency decreases due to more frequent closed GOPs in the elementary stream. In addition, the number of files/requests to be handled is higher, and finally, with shorter segments, TCP throughput may be such that not the full available capacity on the link can be exploited. Annex B.4 and clause 4.3.3.2.2 provide some guidelines on this.
- If files are available in chunks on the origin, for example due to specific encoding or delivery matters, chunked delivery may be supported. If this feature is offered, then the `@availabilityTimeOffset` attribute may be provided to announce how much earlier than the nominal segment availability the segment can be accessed.
- In order to provide tight synchronization between client and server, and therefore providing the receiver the ability to request the segment at the actual segment availability time, the availability time synchronization as defined in clause 4.7 should be provided and signalled in the MPD. Typically support for `http-xsdate` is sufficient for consistent latency support. Accurate NTP synchronization is recommended, but not required for the MPD packager or the DASH client as long as the time synchronization API is provided.
- It is proposed that a client consistently implements and joins at a segment that is slightly offset (e.g. 4 segments earlier) from the live edge segment. The exact number depends on the distribution system (for example in a fully managed environment, the offset may be smaller in contrast to best effort networks). The MPD author may support consistency by providing a suggested presentation delay in the service offering. For details on joining at the live edge, please refer to clause 4.3.4.4.2.

9.11.2. Unanticipated new periods

An [MPD](#) has a certain duration after download during which the service guarantees that the information within remains valid, signaled by `MPD@minimumUpdatePeriod`. To avoid that the clients take future segment existence for granted even if a sudden change on the service offering is necessary, the MPD service provider must set to the `MPD@minimumUpdatePeriod` to a low value.

In the most conservative case, `[[#live-mup-zero|the MPD author sets the MPD@minimumUpdatePeriod to 0]]`. Then no promise for future segments is provided. The DASH client is forced to revalidate the MPD prior to any new Segment

request.

For controlling future MPD validity, basically two options exist:

1. Client downloads a fresh MPD before every Segment request (or batch of requests), preferably using a [conditional GET](#) in order to avoid unnecessary downlink traffic and processing in the client.
2. Client relies on [MPD validity expiration events in event messages](#), if content provider announces those in the MPD and by this, it can revalidate.

The two methods are not mutually exclusive.

9.11.3. Media segment duration variations

Variable [media segment](#) durations need to be correctly signed in the [MPD](#). The mechanism depends on the [addressing mode](#):

1. [Simple addressing](#) allows for [a deviation of up to 50% segment duration in segment start points](#), allowing for some drift to be compensated.
 - If the DASH packager receives a segment stream such that the drift can no longer be compensated, then a new [period](#) SHALL be started, adjusting the addressing parameters to compensate. The [representations](#) SHOULD also be signaled as [period-connected or period-continuous](#).
2. [Explicit addressing](#) allows the duration of each [media segment](#) to be defined explicitly.

[Media segments](#) SHALL NOT have a duration greater than `MPD@maxSegmentDuration` in any situation.

9.11.4. Losses and operational failures

One of the most complex aspects are occasional operational issues, such as losses, outages, failovers of input streams, encoders, packagers and distribution links. Section 4.8 provides detailed overview on available tools that should be used by network service offering and clients in order to deal with operational issues. Several types of losses may occur:

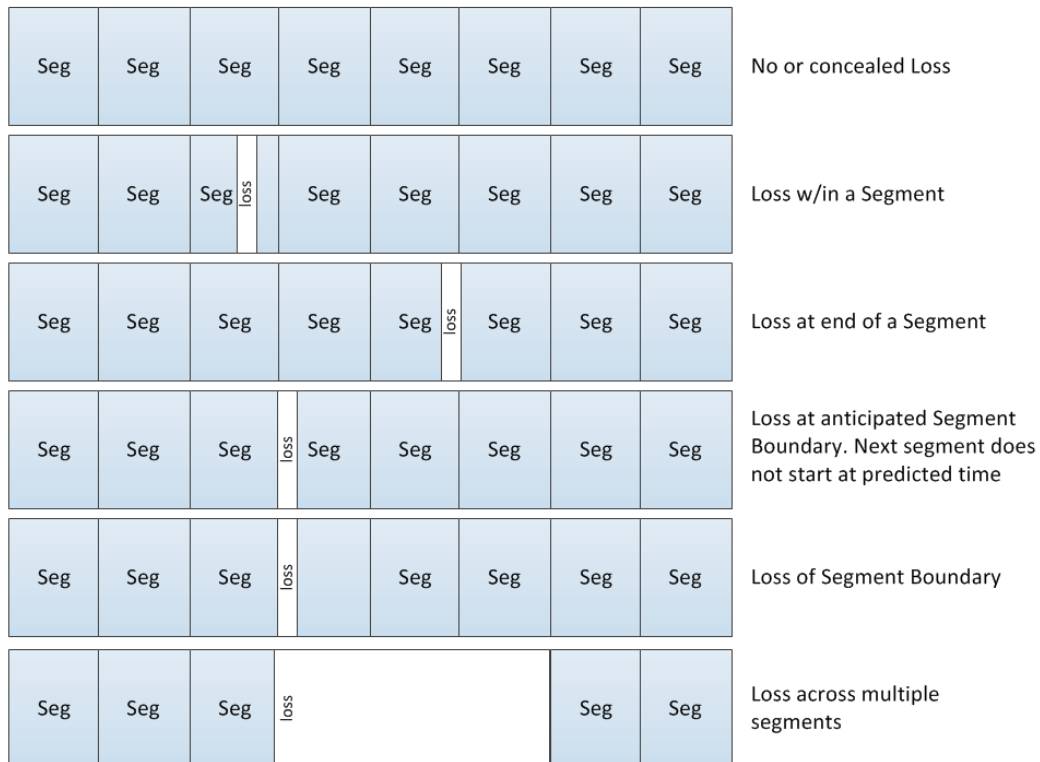


Figure 38 Examples of different types of data loss.

Losses may occur in the middle of a Segment, at the end of a Segment, at the start of a new Segment. At the elementary stream level, losses may be within a compressed access unit (AU), producing a syntactically corrupt bitstream, or may be the result of the ABR encoder simply not encoding a source frame in which case the duration of the prior AU is extended producing a conforming bitstreams. Losses may impact an entire Segment or may just

impact a part of the Segment. Typically, service oriented losses will occur until the next Random access point, i.e. a loss is to be signaled from the start of the lost sample up to the next random access point, typically coinciding with the start of a new Segment.

IOP defines some basic constraints in the [timing model](#):

- [Periods](#) are covered by [media segments](#) in their entirety.
- [Periods](#) are covered by media samples in their entirety.

Deviation from these constraints is not allowed, even in case of data loss. This means that there are basically two options:

1. A service MAY replace lost data with padding data.
2. A service MAY start a new period when the data loss starts and ends, removing the affected [representations](#) for the duration of the loss.

Of course, it is not possible for a service to compensate for data loss in the CDN layer. Clients are expected to survive arbitrary 404 errors that occur due to CDN faults, either by retrying, switching to another CDN (base URL), switching to another [representation](#) or automatically seeking forward.

ISSUE 26 Is there something that goes into more depth about 404s? These statements need a better home.

9.11.5. Minimizing MPD updates

MPD updates, the frequency of MPD updates and the actions included in MPD updates are different ones, and their effects may have different impacts on deployments. To avoid confusion on the generally overloaded term, some more details are discussed in the following section. In non-DASH adaptive streaming solutions, MPD updates result in the following additional processing and delivery overhead:

1. The client sends an uplink requests for the MPD. At least from a CDN perspective, this is issue is considered less critical, typically the bounds of operation are reached by throughput, not by the number of requests.
2. The server needs to send a full MPD with every request, which for itself causes overhead from all the way of the origin server to the client. This is in particular relevant if the manifest contains a list of URLs, and some timeshift buffer is maintained.
3. Yet another aspect is the regular parsing and processing of the manifest in the client. Whereas the processing is likely less of a burden, the consistency across two parsing instances is relevant and requires to keep state.
4. MPD updates may also result in writing a new MPD on the server. This may be less problematic for certain cases, especially for unicast, but it results in significant overhead if DASH formats are used for broadcast.

DASH-IF IOP provides different means to avoid one or the more of the above issues. Assuming that the `MPD@minimumUpdatePeriod` is set to a low value for reasons documented above, then issues mentioned above can be addressed by the following means in DASH-IF IOP:

1. Client Requests: can be avoided by signalling inband that an MPD is has expired. The most obvious tool is the use of Inband Events with MPD expiry. However, this requires inband events being added during packaging.
2. Sending Full MPD: Instead of requesting the full MPD, the client can support this operation by issuing a conditional GET. If the MPD has not changed, no MPD needs to be sent and the downlink rate is small. However, this requires the usage of `@duration` or `SegmentTimeline` with `@r=-1`.
3. MPD Parsing and Processing: This can be avoided by using either of the solutions documented above.
4. MPD writing on server: This goes hand-in-hand with 2, i.e. the usage of `@duration` or `SegmentTimeline` with `@r=-1`.

Generally, DASH-IF IOP provide several tools to address different aspects of minimizing MPD updates. Based on the deployment scenario, the appropriate tools should be used. However, it is preferable that DASH clients support different tools in order to provide choices for the service offering.

9.11.6. Proposed service configuration and MPD generation logic

The core concept is the availability of a segment stream at the input to a packager. The segment stream may be made available as individual segments or as boundary markers in a continuous stream. In addition, the stream may contain information that is relevant for the packager, such as program changes. The segment stream determines for each segment the earliest presentation time, the presentation duration, as well as boundaries in program offerings.

Furthermore, it is assumed that multiple bitrates may exist that are switchable. In the following we focus on one

segment stream, but assume that in the general case multiple bitrates are available and the encoding and segment streams are generated such that they can be switched.

The high-level assumptions for the service are summarized in 4.11.2. Based on these assumptions, a more detailed model is provided.

- A segment stream is provided for each Representation. The segmentation is the same for Representations that are included in one Adaptation Set. Each segment i has assigned a duration $d[i]$ and an earliest presentation time $ept[i]$. In addition, the segment stream has a nominal segment duration d_0 that the ABR encoders attempts to maintain. However, variation may occur for different reasons, documented above.
- Losses may occur in the segment stream, spanning a part of a segment, multiple segments, a full segment and so on. The loss may be in one Representation or in multiple Representations at the same time (see above for more discussions).
- The latency of the time that the segment is made available to the DASH packager and that it is offered as an available segment in the MPD should be small, i.e. the segment availability time should be shortly after the time when the full segment is received in the DASH packager. Any permitted delay by the MPD Packager can be view as additive to change lead time and may therefore improve efficiency and robustness, but may at the same time increase the end-to-end latency.
- Changes in the program setup may occur, that signal changes as discussed in 4.11.2. A change is possibly announced with a time referred to as change lead time. Note that signal changes such as SCTE-35 only indicate where a change may occur, it does not indicate what type of change will occur.

The different scenarios are summarized in Figure 10. For the third part, it shows the notion of the change lead time. Segment of Period with index j are provided. In this case, at the start of segment $(j, i+1)$ (i.e. its earliest presentation time) an indication is provided that the media will change after segment $(j, i+2)$, i.e. the change lead time is $d[j, i+1] + d[j, i+2]$. A new Period $j+1$ is generated that starts with a new segment numbering.

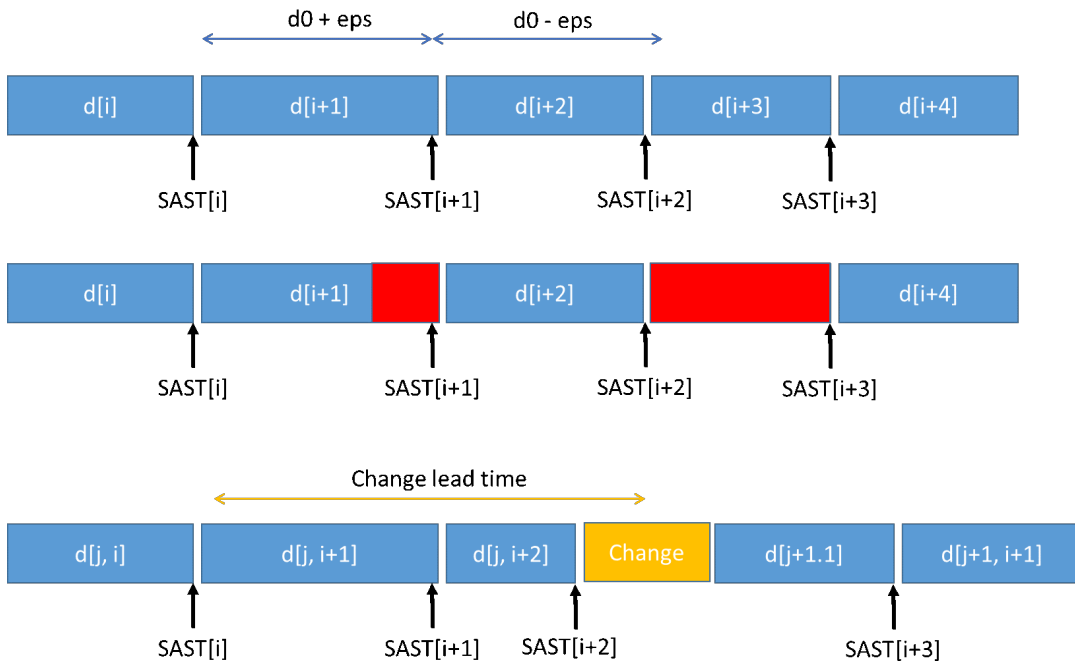


Figure 39 Different properties of a segment stream.

Based on the discussions in 4.11.2, proposed service configuration for such a service are proposed. The service configuration differentiates two deployment scenarios:

1. Clients implementing the simple live client, i.e. no emsg support and no segment parsing is implemented.
2. Clients implementing the main client, i.e. emsg is supported and segment parsing is implemented.

9.11.6.1. Service configuration for simple live

Assuming that the input stream is a segment stream with the properties documented above is received by the DASH packager.

The DASH packager may operate as follows:

- The @minimumUpdatePeriod is set to a value that is equal or smaller than the change lead time provided by the segment stream.
- The @timescale of the Adaptation Set is set to the timescale of the included media
- The @duration attribute is set such that the nominal duration d0 is documented in the MPD for this Adaptation Set.
- \$Number\$ is used of segment templating.
- With incoming segments of the segment stream, a new segment is generated by the DASH packager and the DASH packager checks the validity of the MPD offering. If still valid, no changes to MPD are done. Only if changes are done that are no longer valid, a new MPD is written. Specifically,
 - The [MPD start time](#) of the next segment must be in the range of $EPT - PTO - 0.5DUR$ and $EPT - PTO + 0.5DUR$ with DUR the value of @duration.
 - If this is not fulfilled a new Period is written that includes the following:
 - The Period@start is set such that the MPD start time is correct.
 - The @presentationTimeOffset is set to the EPT of the first segment
 - The @startNumber is set to the first segment in the new Period.
 - The Adaptation Sets are continued by providing Period continuity signalling with each Adaptation Set.
- when an encoder fails for one or more specific Representations to generate the next segment, then the DASH content generator
 - terminates the Segment with the last sample in the segment, (which is possibly corrupted)
 - generates a new MPD as follows:
 - The @minimumUpdatePeriod is set to 0.
 - If all or at least many Representations fail, the Period@duration is set to the value of the media time in the Period that is still available.
 - If only a subset of the Representations fail, the @presentationDuration for the last segment is set to the value of the last presentation time in the Representation that is still available.
 - By doing so, the content provider basically informs the DASH client that for the duration of the Segment as announced, no media is available. The DASH client revalidates this after every Segment duration. The MPD is not changed on the server until either the decoder resumes or the Media Presentation is terminated.
 - If the @minimumUpdatePeriod is long, then the client may request non-existent segments, which itself may then trigger that the DASH client revalidates the MPD. If the DASH client has the possibility, it should add the 'lmsg' brand as a compatibility brand to the last generated segment. In addition, when the segment is distributed over HTTP, the HTTP header should signal the content type of the segment including the compatibility brand 'lmsg'. If the DASH client can identify this, it is expected to refetch the MDP and may by this means observe the early terminated Period or Representations.
 - Only after the encoder resumes, a new MPD is written as follows:
 - A new Period is provided with Period@start according to the value of the new Period. The @presentationTimeoffset of the Representation of the Period shall match the the earliest presentation time of the newly generated Segment. If appropriate, Period connectivity should be signaled.
 - The @minimumUpdatePeriod is set again to the minimum change lead time.
- when a program change is announced, generates a new MPD as follows:
 - The @minimumUpdatePeriod is set to 0.
- When the program change occurs
 - Write a new MPD with all the parameters
 - Reset the @minimumUpdatePeriod is set to a value that is equal or smaller than the change lead time provided

9.11.6.2. Service configuration for main live

Assuming that the input stream is a segment stream with the properties documented above is received by the DASH packager.

The DASH packager may operate as follows:

- The @minimumUpdatePeriod is set to 0.
- The @timescale of the Adaptation Set is set to the timescale of the included media
- The segment timeline is used. Addressing may used: \$Number\$ or \$Time\$.
- The MPD is assigned an MPD@publishTime
- With incoming segments of the segment stream, following the rules in 4.5.2.2 the DASH Packager uses the Segment Timeline to accurately signal the different segment durations. If the segment duration changes, then the @r attribute of the last S element in the Segment timeline is terminated and a new S element is added to the MPD with the new segment duration. The values @t and @d need to be set correctly:
 - @r of the last segment element may be set to -1. In this case a new MPD is only written if the segment duration changes
 - @r of the last segment element may be set to the actual published number of segments. In this case a new MPD is written for each new segment
- Whenever a new MPD is written, the MPD@publishTime is updated.
- when an encoder fails for one or more specific Representations to generate the next segment, then the DASH packager
 - terminates the Segment with the last sample in the segment (may be corrupt)
 - adds emsg to this last generated segment. The MPD validity expiration is set to the duration of the current segment or smaller. This emsg may be added to all Representation that have observed this failure, to all Representations in the Adaptation Set or to all Representations in the MPD. The content author should be aware that if the emsg is not signaled with all Representations, then there exist cases that a switch to the erroneous Representation causes a request to a nonexisting Segment. That loss would be signaled in the MPD, but the client is not aware that an update of the MPD is necessary.
 - The emsg shall be added to all Representations that announce that they carry the message as an inband stream.
 - The MPD is updated on the server such that the last generated segment is documented in the Segment timeline and no new S element is added to the timeline.
 - Only after the Representation(s) under loss resumes, a new S element is written with S@t matching the earliest presentation time of the newly generated Segment. The DASH client with it next update will resume and possibly take into account again this Representation.
 - If the encoder does not resume for a specific Representation over a longer time, it is recommended to terminate this Period and remove this Representation at least temporarily until the encoder resumes again. Period continuity should be signaled.
- when the program change occurs
 - adds emsg to this last generated segment. The MPD validity expiration is set to the duration of the current segment or smaller. This emsg shall be added to all Representations that announce the Inband Event stream for the MPD validity expiration.
 - Write a new MPD with all the parameters
- Whenever a new MPD is written, the MPD@publishTime is updated.

The DASH client having received an MPD that signals gaps is expected to either look for alternative Representations that are not affected by the loss, or if not possible, do some appropriate error concealment. The DASH client also should go back regularly to check for MPD updates whether the Representation gets available again.

10. Ad insertions

ISSUE 27 Needs to be checked for conformance with timing model.

ISSUE 28 Needs proper Bikeshed formatting and referencing

ISSUE 29 Needs deduplication of DASH concepts that are re-defined here.

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.

The following definitions are used in this section:

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 an ad break, pending switch to the 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, encapsulator, or segmenter. Packager does not communicate directly with the origin server – its output is written to the origin server’s storage.

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 its stream may be switched to the stream of another content, e.g. to an ad.

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.

DASH ad insertion relies on several DASH tools defined in [\[MPEGDASH\]](#), which are introduced in this section. The correspondence between these tools and ad insertion concepts are explained below.

10.1. 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.

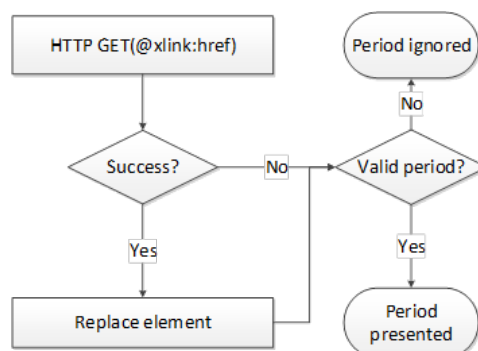


Figure 40 XLink resolution

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 with `@xlink:actuate` set to `onRequest`, 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.

Note: In ISO/IEC 23009-1:2014/Cor.3 it is clarified that if multiple top-level remote elements are included, the remote element entity is not a valid XML document.

10.2. Periods§

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 $\text{PeriodStart}(i+1) - \text{PeriodStart}(i)$. 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 $\text{PeriodStart}(i+1)$.

`Representation@presentationTimeOffset` specifies the value of the presentation time at $\text{PeriodStart}(i)$.

10.3. 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.

10.4. 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 or connectivity should be used and signaled as defined in section 3.2.12 and ISO/IEC 23009-1:2014/Amd.3 [4].

10.5. 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).

10.6. 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 [54])

10.7. 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. For 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.

10.8. 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 [41], cookies set via 2xx, 4xx, and 5xx responses must be processed and have explicit timing and security model.

10.9. 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 5.3.3.7.

DASH Callback events are defined in ISO/IEC 23009-1:2014 AMD3 [4], 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. Callback events can be both, MPD and inband events.

10.10. Ad insertion 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).

10.11. Server-based architecture

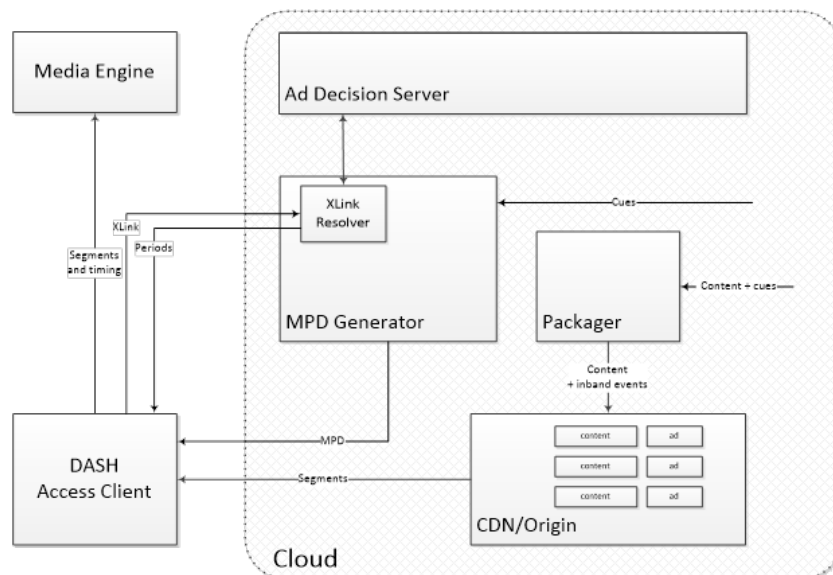


Figure 41 Server-based architecture

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 5.1.3.5.

In the live case, packager receives feed containing inband cues, such as MPEG-2 TS with SCTE 35 cue messages [54]. The packager ingests content segments into the CDN. 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.

10.11.1. Implementation basics

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 than 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.

10.11.2. 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. 5.1.3.5 and 5.1.3.4). 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.

10.11.3. Timing and dereferencing

The only interface between DASH client and the XLink resolver is the XLink URL (i.e., the `Period@xlink:href` attribute). After resolution, the complete remote Period element is replaced with Period element(s) from the remote entity (body of HTTP response coming from XLink resolver). This means that the XLink resolver is (in the general case) unaware of the exact start time of the ad period.

In case of linear content, start of the ad period is only known a short time before the playback. The recommended implementation is to update the MPD at the moment the start of the ad period is known to the MPD generator.

The simplest approach for maintaining time consistency across dereferencing is to have the MPD update adding a `Period@duration` attribute to the latest (i.e., the currently playing) main content period. This means that the MPD resolver needs to include the `Period@duration` attribute into each of the Period elements returned in the remote entity. The downside of this approach is that the DASH client needs to be able to update the currently playing period.

An alternative approach is to embed the desired value of `Period@start` of the first period of the remote entity in the XLink URL (e.g., using URL query parameters). This approach is described in clause 5.3.5. The downside of this alternative approach is that the DASH specification does not constrain XLink URLs in any way, hence the XLink resolver needs to be aware of this URL query parameter interface defined in clause 5.3.5.

10.11.4. 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 or connectivity should be signaled, if the content obeys the rules.

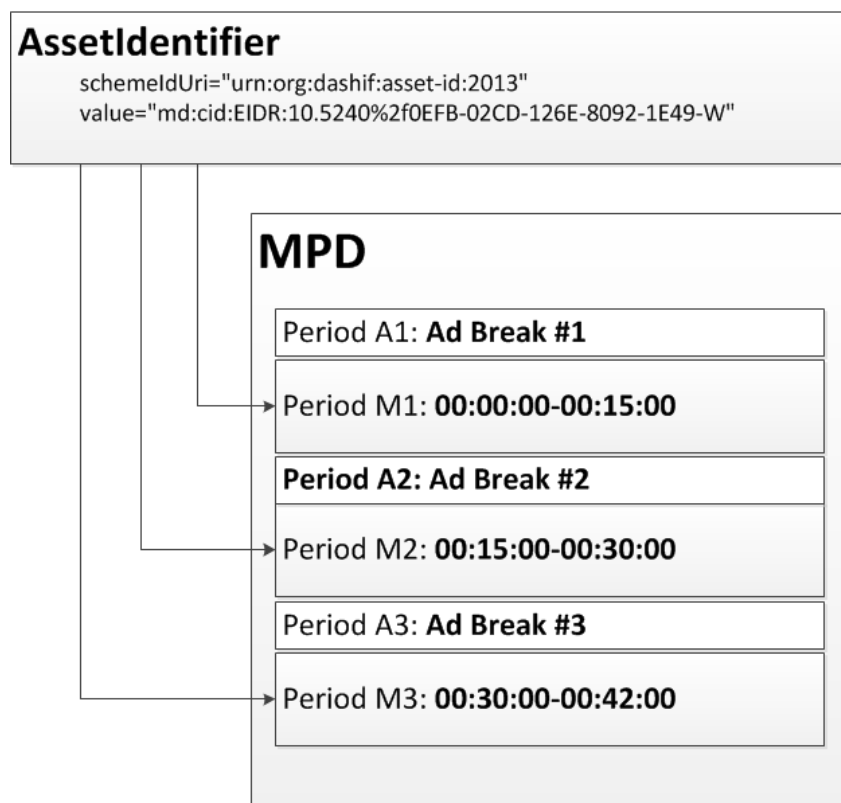


Figure 42 Using an asset identifier

10.11.5. 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.

10.11.6. 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 `@schemeldUri` provides us with such addressing functionality, while MPD events allow us to put opaque payloads into the MPD.

10.11.7. Workflows§

In the workflows below we assume that our inputs are MPEG-2 transport streams with embedded SCTE 35 cue messages [54]. 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.

10.11.8. Linear workflow§

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.

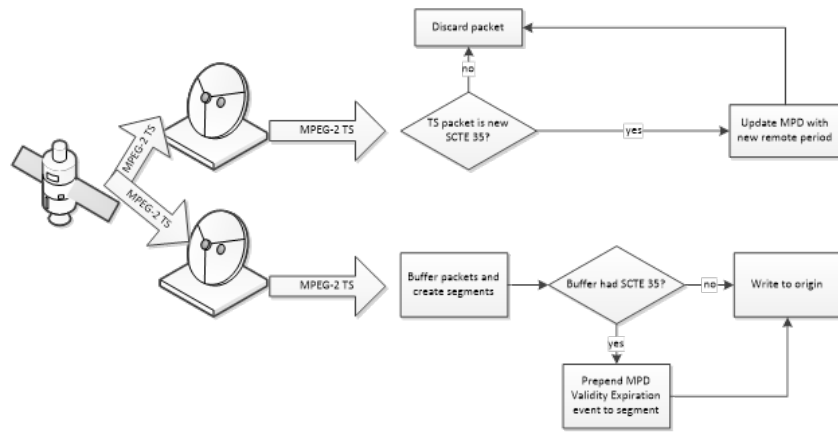


Figure 43 Live workflow

10.11.8.1. 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.

EXAMPLE 13

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.

EXAMPLE 14

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 5.1.3.6), 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

EXAMPLE 15

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 [60], 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 [55], but uses `base64url` and not `base64` encoding as the section is included in a URI.

10.11.8.2. 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.

10.11.8.3. 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.

10.11.8.4. Cancellation

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

When cancellation 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.

10.11.8.5. 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.

10.11.8.6. 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.

10.11.8.7. Ad decision

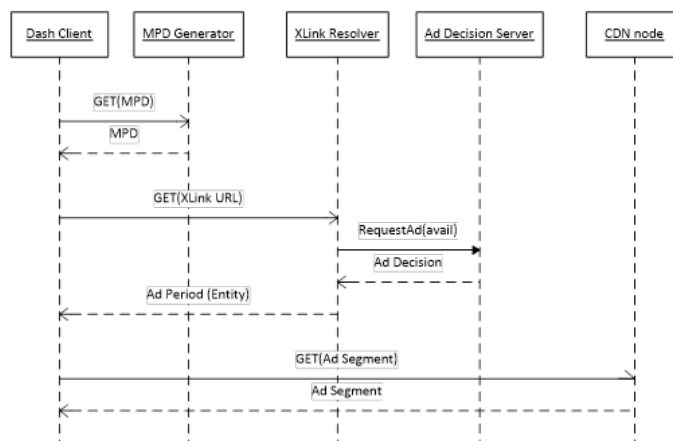


Figure 44 Ad decision

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 5.3.3.2.1.3, 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 5.3.3.2.1.1 "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.

10.11.9. On demand workflow

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 5.3.3.2) 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 5.3.3.2 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.

10.11.9.1. 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 5.3.3.5 below.

10.11.9.2. 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.

10.11.9.3. 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.

10.11.9.4. 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 [53]. 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. VAST 3.0 response can be carried in an Event element where EventStream@schemeldUri value is http://dashif.org/identifiers/vast30.

An alternative implementation uses DASH Callback events to point to the same tracking URLs. While DASH specification permits both inband and MPD Callback events, inband callback events shall not be used.

10.11.10. Examples

EXAMPLE 16

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.

```
<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>

  <Period start="PT0.00S" duration="PT600.65" 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="$Bandwidth%/init.mp4v"
        media="$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>

  <Period duration="PT60.65" id="ad break #1"
    xlink:href="https://adserv.com/avail.mpd?scte35-time=PT600.65&
cte35-cue=DAIAAAAAAAAAAAQAAZ_I0VniQAQAgBDVUVJQAAAAH+cAAAAA%3D%3D"
    xlink:actuate="onRequest" >

    <AdaptationSet mimeType="video/mp4" codecs="avc1.640828"
      frameRate="30000/1001"
      segmentAlignment="true" startWithSAP="1">
      <BaseURL availabilityTimeOffset="INF">default_ad/</BaseURL>
      <SegmentTemplate timescale="90000" initialization="$Bandwidth%/init.mp4v"
        media="$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="PT600.65" 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="$Bandwidth%/init.mp4v"
    media="$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>

<Period duration="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" >

  <AdaptationSet mimeType="video/mp4" codecs="avc1.640828"
    frameRate="30000/1001"
    segmentAlignment="true" startWithSAP="1">
  <BaseURL availabilityTimeOffset="INF">default_ad2/</BaseURL>
  <SegmentTemplate timescale="90000" initialization="$Bandwidth%/init.mp4v"
    media="$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>

```

10.11.11. Use of query parameters

Parameters can be passed into the XLink resolver as a part of the XLink URL. Clause 5.3.3.2.1.3 shows an example of this approach when an SCTE 35 cue message is embedded into the XLink URL.

This approach can be generalized and several parameters (i.e., name-value pairs) can be defined. SCTE 214-1 2016 [56] takes this approach and defines parameters expressing splice time (i.e., Period@start of the earliest ad period), SCTE 35 cue message, and syscode (a geolocation identifier used in US cable industry). The first two parameters are also shown in example in clause 5.3.4.1 of this document.

Note: Effectively this creates a RESTful API for XLink dereferencing. While discussion above implies that these parameters are embedded by the MPD generator into the XLink URL, the parameter values may as well be calculated by the client or the embedded values may be modified by the client.

Note: The same RESTful API approach can be used with MPD URLs as well.

Note: More parameters may be defined in the future version of these guidelines.

10.12. App-based architecture

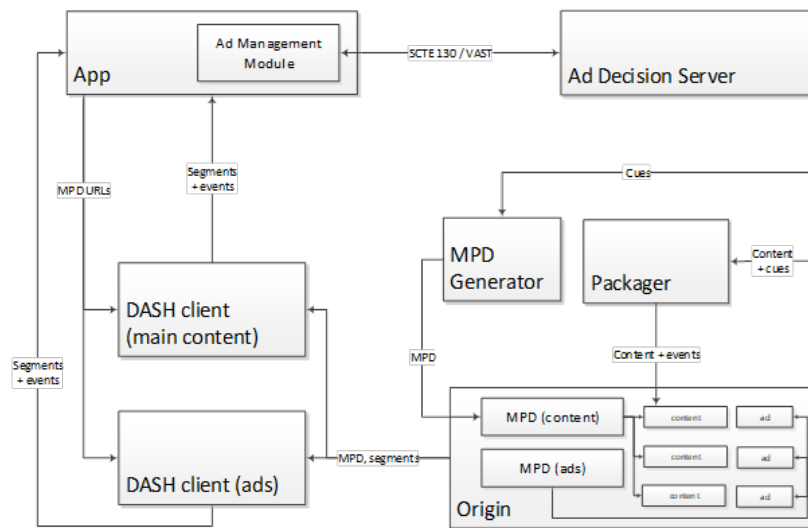


Figure 45 App-based architecture

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.

10.12.1. Implementation basics

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.

10.12.2. SCTE 35 events

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 18 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 214-3 [57].

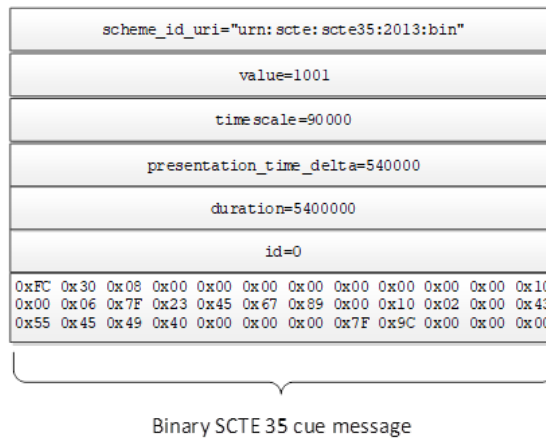


Figure 46 Inband carriage of SCTE 35 cue messages

Figure 19 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:scte35:2014:xml+bin">
  <Event timescale="90000" presentationTime="54054000" duration="5400000" id="1">
    <scte35:Signal>
      <scte35:Binary>
        /DAIAAAAAAAAAAAQAAZ/I0VniQAQAgBDVUVJQAAAAH+cAAAAA==
      </scte35:Binary>
    </scte35:Signal>
  </Event>
</EventStream>
```

Figure 47 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 ANSI/SCTE 214-1 [56] sec 6.8.4 (MPD) and SCTE 214-3 [57] sec 8.3.3.

10.12.3. Asset identifiers

See sec. 5.3.2.2 for details.

10.12.4. Linear workflow

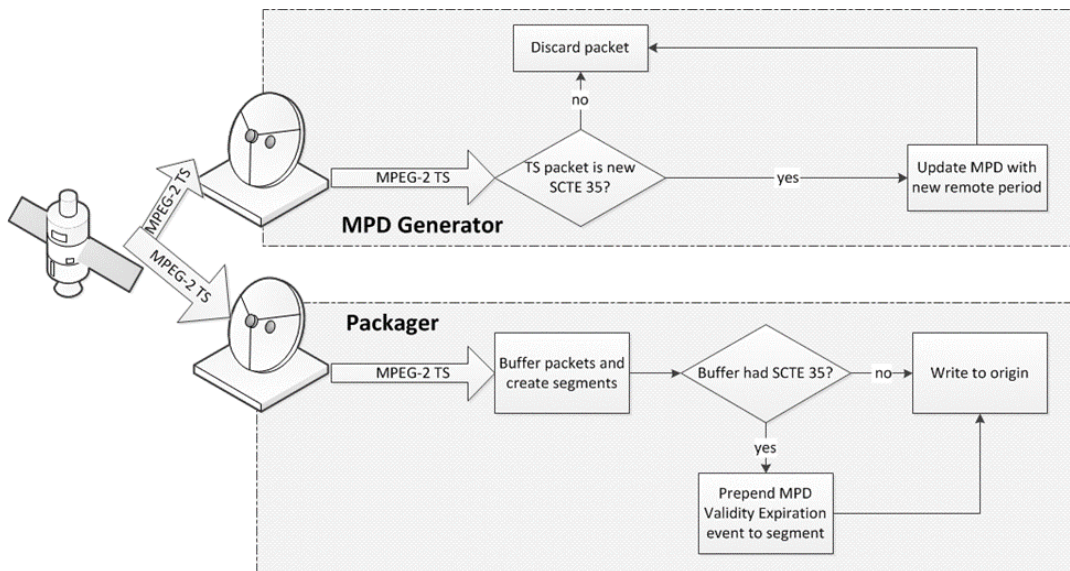


Figure 48 Linear workflow for app-driven architecture

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

10.12.5. On demand workflow

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.

10.13. AssetIdentifier extensions

ISSUE 30 What are "extensions"? Move this to features/constraints chapters?

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 and may be used for inserted content. In order to enable better tracking and reporting,

unique IDs should be used for different assets.

Use of EIDR and Ad-ID identification schemes is recommended. The value of @schemeldUri set to "urn:eidr" signals use of EIDR. The value of @value attribute shall be a valid canonical EIDR entry as defined in [67].

Use of Ad-ID for asset identification is signaled by setting the value of @schemeldUri to "urn:smp:ul:060E2B34.01040101.01200900.00000000" ("designator" URN defined in SMPTE 2092-1 [68]). The value of @value attribute shall be a canonical full Ad-ID identifier as defined in SMPTE 2092-1 [68].

Other schemes may be used, including user private schemes, by using appropriately unique values of @schemeldUri.

In the absence of other asset identifier schemes, a DASH-IF defined scheme may be used with the value of @schemeldUri set to "urn:org:dashif:asset-id:2014". If used, the value of @value attribute descriptor shall be a MovieLabs ContentID URN ([58], 2.2.1) for the content. It shall be the same for all parts of an asset. Preferred schemes are EIDR (main content) and AdID (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.

10.14. Remote period extensions§

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.

Period@xlink:actuate="onRequest" shall not be used if MPD@type="dynamic" 5

10.15. User-defined event extensions§

10.15.1. Cue message§

Cue messages used in app-driven architecture shall be SCTE 35 events [54]. SCTE 35 event carriage is defined in ANSI/SCTE 214-1 (MPD) and ANSI/SCTE 214-3 (inband). For MPD events, the XML schema is defined in SCTE 35 2014 [54] 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.

10.15.2. Reporting§

MPD events with embedded IAB VAST 3.0 [53] 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 [4].

10.15.3. 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 ANSI/SCTE 214-3.
2. "urn:scte:scte35:2014:xml+bin" for SCTE 35 MPD events containing only base64 cue message representation, as defined in ANSI/SCTE 214-1. 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. "http://dashif.org/identifiers/vast30" for MPD events containing VAST3.0 responses [53].

4. urn:mpeg:dash:event:callback:2015 for DASH callback events.

11. Media coding technologies

This chapter describes the constraints that apply to media codecs when used in interoperable services.

Services SHALL use only the media codecs described in this chapter, in conformance with the requirements defined here.

Clients MAY support any set of codecs described in this chapter and SHALL NOT attempt to play back [representations](#) for which they do not have codec support.

11.1. H.264 (AVC)

The H.264 (AVC) codec [\[MPEGAVC\]](#) MAY be used by services for [video adaptation sets](#). Clients SHOULD support this codec.

For [representations](#) up to 1280x720p resolution and up to 30 fps, the H.264 (AVC) Progressive High Profile Level 3.1 decoder SHALL be used.

For [representations](#) up to 1920x1080p resolution and up to 30 fps, the H.264 (AVC) Progressive High Profile Level 4.0 decoder SHALL be used.

The encapsulation of H.264 data in DASH containers SHALL conform to [\[iso14496-15\]](#).

Clients SHALL support SPS/PPS storage both in the [initialization segment](#) (sample entry `avc1`) and inband storage (sample entry `avc3`). Services MAY use either form.

Note: Use of `avc3` is one of the factors that enables [bitstream switching](#).

The below table lists examples of `@codecs` strings for H.264 (AVC) that match the decoders defined in this chapter.

Profile	Level	@codecs
H.264 (AVC) Progressive High Profile	3.1	<code>avc1.64Y01F</code>
		<code>avc3.64Y01F</code>
	4.0	<code>avc1.64Y028</code>
		<code>avc3.64Y028</code>

Figure 49 Example `@codecs` strings for H.264 (AVC)

Note: Other `@codecs` strings may also be compatible (a higher level decoder can typically decode content intended for a lower level decoder).

For a detailed description on how to derive the signaling for the codec profile for H.264/AVC, see [\[DVB-DASH\]](#) section 5.1.3.

11.2. H.265 (HEVC)

The H.265 (HEVC) codec [\[MPEGHEVC\]](#) MAY be used by services for [video adaptation sets](#).

For [representations](#) up to 1280x720p at up to 30 fps, the HEVC Main Profile Main Tier Level 3.1 decoder SHALL be used.

For [representations](#) up to 2048x1080 at up to 60 fps at 8-bit frame depth, the HEVC Main Profile Main Tier Level 4.1 decoder SHALL be used.

For [representations](#) up to 2048x1080 at up to 60 fps at 10-bit frame depth, the HEVC Main10 Profile Main Tier Level 4.1 decoder SHALL be used.

The encapsulation of H.265 data in DASH containers SHALL conform to [\[iso14496-15\]](#).

Clients SHALL support VPS/SPS/PPS storage both in the [initialization segment](#) (sample entry `hvc1`) and inband storage (sample entry `hev1`). Services MAY use either form.

Note: Use of hev1 is one of the factors that enables [bitstream switching](#).

ISSUE 31 Where does UHD fit? Why is it in a separate chapter? We should unify.

The [\[ISOBMFF\]](#) sync sample signaling and [\[MPEGDASH\]](#) SAP type signaling SHALL be derived from the following table.

NAL unit type	[ISOBMFF] sync sample flag	[MPEGDASH] SAP type
IDR_N_LP	true	1
IDR_W_RADL	true	2 (if the IRAP has associated RADL pictures) 1 (if the IRAP has no associated RADL pictures)
BLA_N_LP	true	1
BLA_W_RADL	true	2 (if the IRAP has associated RADL pictures) 1 (if the IRAP has no associated RADL pictures)
BLA_W_LP	false	3 (if the IRAP has associated RASL pictures)
	true	2 (if the IRAP has no associated RASL pictures but has associated RADL pictures)
	true	1 (if the IRAP has no associated leading pictures)
CRA	false	3 (if the IRAP has associated RASL pictures)
	true	2 (if the IRAP has no associated RASL pictures but has associated RADL pictures)
	true	1 (if the IRAP has no associated leading pictures)

Figure 50 Signaling dependent on HEVC IRAP pictures in [\[ISOBMFF\]](#) and [\[MPEGDASH\]](#).

IOP requires that each [media segment](#) start with SAP type 1 or 2. If the above table indicates SAP type 3, the content is not conforming to IOP.

When the table above lists multiple possible values for a given NAL unit type and the entity creating the signaling is not able to determine correctly which values to use, it SHALL use the first value listed in the table for that NAL unit type.

The below table lists examples of @codecs strings for H.265 (HEVC) that match the decoders defined in this chapter.

Profile	Level	@codecs
HEVC Main	3.1	hev1.1.2.L93.B0 hvc1.1.2.L93.B0
	4.1	hev1.1.2.L123.B0 hvc1.12.L123.B0
HEVC Main-10	4.1	hev1.2.4.L123.B0 hvc1.2.4.L123.B0

Figure 51 Example @codecs strings for H.265 (HEVC)

Note: Other @codecs strings may also be compatible (a higher level decoder can typically decode content intended for a lower level decoder).

For a detailed description on how to derive the signaling for the codec profile for H.265/HEVC, see [\[DVB-DASH\]](#) section 5.2.2.

11.3. Decoder configuration with H.264 and H.265§

This chapter applies only to [video adaptation sets](#) that use H.264 or H.265.

All [initialization segments](#) in the same [video adaptation set](#) SHALL use the same sample description (i.e. no mixing of avc1 and avc3 is allowed).

In [representations](#) using avc1 or hvc1 sample description:

- All decoding parameter sets referenced by NALs 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](#).
- Edit lists MAY be present.

In [representations](#) using avc3 or hev1 sample description:

- All decoding parameter sets referenced by NALs SHALL be indexed to a Sequence Parameter NAL (SPS) and Picture Parameter NAL (PPS) stored prior to the first video sample in the same [media segment](#).
- SPS and PPS stored in each [media segment](#) SHALL be used for decoding and display scaling.
- Every [initialization segment](#) SHALL include an avcC or hvcC box that SHALL include SPS and PPS NALs that equal the highest Tier, Profile, Level and vertical/horizontal sample count of any SPS in the [representation](#).
 - HEVC Decoder Configuration Records shall also include a VPS NAL.
- SPS and PPS stored in the [initialization segments](#) SHALL be used only for decoder and display initialization.
- Edit lists MAY be present if using [indexed addressing](#). Edit lists SHALL NOT be present when using any other [addressing mode](#).

11.4. Bitstream switching with H.264 and H.265§

This chapter applies only to [bitstream switching adaptation sets](#) that use H.264 or H.265.

All [representations](#) SHALL be encoded using the avc3 or hev1 sample description.

The first presented sample's composition time SHALL equal 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.

ISSUE 32 What is the correct scoping for the above requirement? Is the composition time requirement specific to H.264/H.265? Or does it apply to all bitstream switching video? Or does it apply to all bitstream switching, not only video?

11.5. Thumbnail images§

This chapter defines constraints for [thumbnail adaptation sets](#).

[Media segments](#) SHALL be either JPEG or PNG images, using @mimeType of image/jpeg or image/png.

The [adaptation set](#) SHALL carry an [essential property descriptor](#) with @schemeIdUri="http://dashif.org/guidelines/thumbnail_tile". The @value SHALL indicate the number of thumbnails in each [media segment](#), with the syntax being HxV, where:

- H is the number of horizontal tiles in the grid.
- V is the number of vertical tiles in the grid.

Descriptive attributes on the [representation](#) SHALL describe an entire grid of thumbnails (one [media segment](#)), not an individual thumbnail.

Note: JPEG images have a maximum width and height of 64K pixels.

Thumbnails stored in one grid SHALL be evenly distributed in time across the time span covered by the [media segment](#) on the [MPD timeline](#), from left to right, then top to bottom.

EXAMPLE 17

Thumbnail presentation order for a 3x3 grid:

```
1 2 3
4 5 6
7 8 9
```

EXAMPLE 18

The following [thumbnail adaptation set](#) defines one [media segment](#) for every 125 seconds, containing a 25x1 image grid (25 columns, 1 row) with each image being 256x180 pixels. The display duration of each thumbnail image is 5 seconds. The single thumbnail [representation](#) requires 10 Kbps of bandwidth on average.

```
<AdaptationSet mimeType="image/jpeg">
  <SegmentTemplate media="thumbnails_${Number$.jpg" timescale="1" duration="125" />
  <Representation bandwidth="10000" width="6400" height="180">
    <EssentialProperty schemeIdUri="http://dashif.org/guidelines/thumbnail_tile" value="25x1" />
  />
</Representation>
</AdaptationSet>
```

Parts of the MPD structure that are not relevant for this chapter have been omitted - this is not a fully functional MPD file.

11.6. HE-AACv2 audio (stereo)

The codec for basic stereo audio support is MPEG-4 High Efficiency AAC v2 Profile, level 2 [\[MPEGAAC\]](#).

Note: HE-AACv2 is also standardized as Enhanced aacPlus in 3GPP TS 26.401.

HE-AACv2 Profile decoder can also decode any content that conforms to:

- MPEG-4 AAC Profile
- MPEG-4 HE-AAC Profile

Therefore, services are free to use any AAC version. Typical clients are expected to play AAC-LC, HE-AAC and HE-AACv2 encoded content.

For content with SBR, i.e. `@codecs=mp4a.40.5` or `@codecs=mp4a.40.29`, `@audioSamplingRate` signals the resulting sampling rate after SBR is applied, e.g. 48 kHz even if the AAC-LC core operates at 24 kHz.

For content with PS, i.e. `@codecs=mp4a.40.29`, the `AudioChannelConfiguration` element signals the resulting channel configuration after PS is applied, e.g. stereo even if the AAC-LC core operates at mono.

The encapsulation of HE-AACv2 data in DASH containers SHALL conform to [\[MP4\]](#).

SAP type SHALL be 1. The `@codecs` string SHALL have a value from the below table.

Profile	@codecs
MPEG-4 AAC Profile [11]	mp4a.40.2
MPEG-4 HE-AAC Profile [11]	mp4a.40.5
MPEG-4 HE-AAC v2 Profile [11]	mp4a.40.29

Figure 52 Permitted HE-AACv2 @codecs values.

To conform to [\[DVB-DASH\]](#), explicit backwards compatible signaling SHALL be used to indicate the use of the SBR and PS coding tools for all HE-AAC and HE-AACv2 bitstreams.

ISSUE 33 What does the above requirement actually mean - what does an implementation have to do?
Unclear right now.

11.7. HE-AACv2 audio (multichannel)§

This chapter extends HE-AACv2 requirements with multichannel scenarios. All constraints defined for the stereo scenario also apply here.

Support for multichannel content is available in the HE-AACv2 Profile, starting with level 4 for 5.1 and level 6 for 7.1. Decoders implementing MPEG-4 HE-AACv2 multichannel profiles are fully compatible with content encoded in conformance to HE-AACv2 stereo requirements defined in [IOP](#).

The content SHOULD be prepared incorporating loudness and dynamic range information into the bitstream also considering DRC Presentation Mode in [jiso14496-3-2009-amd4-2013](#).

Decoders SHALL support decoding of loudness and dynamic range related information, i.e. `dynamic_range_info()` and `MPEG4_ancillary_data()` in the bitstream.

11.8. CEA-608/708 Digital Television (DTV) Closed Captioning§

This chapter defines requirements for interoperable use of CEA-608/708 Digital Television (DTV) Closed Captioning [\[CEA708\]](#) in DASH presentations.

Note: This chapter is compatible with draft SCTE specification DVS 1208 and therefore SCTE URNs are used for the descriptor `@schemeIdUri`.

CEA-608/708 captions SHALL be carried in SEI messages embedded in [representations](#) of a [video adaptation set](#), with the encapsulation as defined in [\[SCTE128-1\]](#), section 8.1. The SEI message `payload_type=4` is used to indicate that Rec. ITU-T T.35 based SEI messages are in use.

ISSUE 34 ITU-T T.35 referenced above seems unrelated to the topic. What is the correct reference?

ISSUE 35 Is the `payload_type` sentence meant to be a requirement or a description of the referenced spec or what is the utility of this statement in IOP?

Every [representation](#) in the [video adaptation set](#) SHALL have identical CEA-608/708 captions. Both CEA-608 and CEA-708 MAY be present simultaneously in the same [video adaptation set](#).

The presence of CEA-608/708 captions SHALL be signaled by an `Accessibility` descriptor on the [adaptation set](#) level, with `@schemeIdUri="urn:scte:dash:cc:cea-608:2015"` or `@schemeIdUri="urn:scte:dash:cc:cea-708:2015"`, with an optional `@value`.

When present for CEA-608 captions, the `@value` of this descriptor SHALL describe the caption streams and languages in conformance to 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 [45]
```

Two variants of `@value` syntax for CEA-608 are described above - a variant with plain language codes and a variant with caption channel numbers. Services SHOULD use the variant with channel numbers.

Note: [IOP](#) does not provide the `@value` syntax for CEA-708. See [\[SCTE214-1\]](#).

EXAMPLE 19

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

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

11.9. Timed Text (IMSC1)§

This chapter defines requirements for using IMSC1 text [61] in DASH presentations.

W3C TTML [\[ttml2\]](#) and its various profiles - W3C IMSC1 [\[ttml-ims1.1\]](#) (text and image profiles), SMPTE Timed Text [\[SMPTE2052-1-2013\]](#), and EBU Timed Text [\[EBU-TT\]](#) - provide a rich feature set for text tracks. Beyond basic

subtitles and closed captioning, for example, graphics-based subtitles and closed captioning are also supported by IMSC1.

Many clients only implement a subset of IMSC1. The exact feature sets used by clients and services may need careful alignment to ensure mutual compatibility. Do not assume that all of IMSC1 is supported by typical clients - this is unlikely.

Conversion of CEA-608 and CEA-708 into IMSC1 SHALL be done according to [\[SMPTE2052-10\]](#) and [\[SMPTE2052-11\]](#), respectively.

One of the following storage formats SHALL be used for IMSC1 [representations](#):

- [\[ISOBMFF\] media segments](#).
- Stand-alone XML file (one file per [representation](#)).

The ISO BMFF encapsulated form SHOULD be used, as stand-alone XML file storage has significant limitations. See also [§ 5.2.10 Timing of stand-alone IMSC1 and WebVTT text files](#).

Note: [\[DVB-DASH\]](#) only supports the ISO BMFF encapsulated form.

The signaling in the [MPD](#) SHALL conform to the below table.

Codec	Storage	@mimeType	@codecs
IMSC1 Timed Text [61]	Stand-alone XML file	application/ttml+xml	
IMSC1 Timed Text [61]	ISO BMFF encapsulation [ISOBMFF] [iso14496-30][29]	application/mp4	See W3C TTML Profile Registry [62]

Figure 53 IMSC1 signaling parameters.

11.10. Enhanced AC-3 (Dolby Digital Plus)§

The @codecs parameter SHALL be ec-3. SAP type SHALL be 1.

The AudioChannelConfiguration element SHALL use @schemeIdUri="tag:dolby.com,2014:dash:audio_channel_configuration:2011" with @value as defined in the [DASH-IF identifier registry](#).

Signaling and encapsulation SHALL conform to [\[ETSI102366\]](#) Annex F.

11.11. Dolby TrueHD§

The @codecs parameter SHALL be m1pa. SAP type SHALL be 1.

Signaling and encapsulation SHALL conform to [\[Dolby-TrueHD\]](#).

11.12. AC-4§

The @codecs parameter SHALL be ac-4. SAP type SHALL be 1.

The AudioChannelConfiguration element SHALL use @schemeIdUri="tag:dolby.com,2014:dash:audio_channel_configuration:2011" with @value as defined in the [DASH-IF identifier registry](#).

Signaling and encapsulation SHALL conform to [\[ETSI103190-1\]](#) Annex E.

11.13. DTS-HD§

DTS-HD [\[ETSI102114\]](#) comprises a number of profiles optimized for specific applications. More information about DTS-HD and the DTS-HD profiles can be found at <https://dts.com/>.

For all DTS formats SAP is always 1.

The signaling and encapsulation SHALL conform to [\[DTS9302J81100\]](#), [\[DTS9302K62400\]](#) and to the below table.

Codec	@codecs
DTS Digital Surround	dtsc
DTS-HD High Resolution and DTS-HD Master Audio	dtsh
DTS Express	dtse
DTS-HD Lossless (no core)	dtsl

Figure 54 DTS @codecs values

11.14. MPEG Surround§

MPEG Surround [\[iso23003-1\]](#) 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.

MPEG Surround used in DASH SHALL comply with level 4 of the Baseline MPEG Surround profile.

SAP type SHALL be 1. @codecs SHALL be mp4a.40.30.

11.15. MPEG-H 3D Audio§

MPEG-H 3D Audio [\[iso23008-3\]](#) encoded content SHALL comply with Level 1, 2 or 3 of the MPEG-H Low Complexity (LC) Profile.

In addition to the requirements in [\[iso23008-3\]](#), the following constraints SHALL apply to storage of raw MPEG-H audio frames in DASH containers:

- One audio ISO BMFF sample shall consist of a single `mpegh3daFrame()` structure, as defined in [\[iso23008-3\]](#) clause 20.5.
- The parameters carried in the `MHADecoderConfigurationRecord()` shall be consistent with the configuration of the audio bitstream. In particular, the `mpegh3daProfileLevelIndication` shall be set to `0x0B`, `0x0C`, or `0x0D` for MPEG-H Audio LC Profile Level 1, Level 2, or Level 3, respectively.
- The `referenceChannelLayout` field carried in the `MHADecoderConfigurationRecord()` shall be equivalent to what is signaled by `ChannelConfiguration` according to [\[iso23001-8\]](#).
- Each `media segment` SHALL start with a SAP of type 1 (e.g. a sync sample). MPEG-H Audio sync samples contain Immediate Playout Frames (IPFs), as specified in [\[iso23008-3\]](#) clause 20.2. For such frames, the raw MPEG-H audio frames shall contain the `AudioPreRoll()` syntax element, as defined in sub-clause 5.5.6 of [\[iso23008-3\]](#), and shall follow the requirements for stream access points as defined in clause 5.7 of [\[iso23008-3\]](#). The `AudioPreRoll()` syntax element carried in the IPFs shall contain a valid configuration structure (`AudioPreRoll.Config()`) and should contain one pre-roll frame (`AudioPreRoll.numPreRollFrames = 1`).

Note: The `mpegh3daConfig()` structure is expected to be different for each [representation](#) in an [adaptation set](#).

SAP type shall be 1.

ISO BMFF encapsulation SHALL conform to [\[iso23008-3\]](#).

Codec	@codecs
MPEG-H 3D audio LC profile level 1	mhm1.0x0B
MPEG-H 3D audio LC profile level 2	mhm1.0x0C
MPEG-H 3D audio LC profile level 3	mhm1.0x0D

Figure 55 Permitted @codecs values

11.16. MPEG-D Unified Speech and Audio Coding§

MPEG-D Unified Speech and Audio Coding (USAC) has been designed to provide consistently high audio quality with a variety of content that comprises a mixture of audio and speech signals. Using such a codec in a DASH streaming environment enables adaptive switching capability from 12 kbps stereo up to transparency.

[\[iso23000-19-2018-amd2-2019\]](#) defines a media profile xHE-AAC for MPEG-D USAC that is suitable for streaming applications.

Usage of USAC in DASH presentations SHALL conform to [\[iso23000-19-2018-amd2-2019\]](#), providing support up to 5.1 multichannel coding.

SAP type SHALL be 1. @codecs SHALL be mp4a.40.42.

11.17. UHD HEVC 4K§

For the support of broad set of use cases the DASH-IF IOP HEVC 4k Extension is defined. UHD HEVC 4k video encoded with H.265/HEVC is an advanced distribution format for TV services that enables higher resolution experiences in an efficient manner.

This extension describes requirements for content at 4k resolutions up to 60fps, and defines the required codec support as HEVC Main 10 Level 5.1.

The conformance to DASH-IF IOP HEVC 4k may be signaled by a @profile attribute with the value <http://dashif.org/guidelines/dash-if-uhd#hevc-4k>.

NAL Structured Video streams conforming to this Media Profile SHALL NOT exceed the following coded picture format constraints:

- Maximum encoded horizontal sample count of 3840 samples
- Maximum encoded vertical sample count of 2160 samples
- Maximum frame rate of 60000 / 1000.

ISSUE 36 There is a bunch of stuff below with no obvious connection to UHD. Should this not also be in the non-UHD HEVC chapter?

Additional coded picture format constraints:

- [Representations](#) in one [adaptation set](#) SHALL only differ by the following parameters: bitrate, spatial resolution, frame rate.
- The condition of the following SHALL NOT change throughout one HEVC video track:
 - `aspect_ratio_idc`
 - `cpb_cnt_minus1`
 - `bit_rate_scale`
 - `bit_rate_value_minus1`
 - `cpb_size_scale`
 - `cpb_size_value_minus1`
- The following fields SHALL NOT change throughout an HEVC elementary stream.
 - `pic_width_in_luma_samples`
 - `pic_height_in_luma_samples`
- YCbCr SHALL be used as the chroma format and 4:2:0 for color sub-sampling.
- The bit depth of the content SHALL be either 8 bit or 10 bit.
- The color primaries SHALL be [\[ITU-R-BT.709\]](#).

The bitstream SHALL comply with the Main10 Tier Main Profile Level 5.1 restrictions as specified in [\[MPEGHEVC\]](#).

UHD HEVC 4k bitstreams SHALL set `vui_parameters_present_flag` to 1 in the active Sequence Parameter Set, i.e. HEVC bitstreams shall contain a Video Usability Information syntax structure.

The sample aspect ratio information shall be signaled in the bitstream using the `aspect_ratio_idc` value in the Video Usability Information (see [\[MPEGHEVC\]](#) table E1). UHD HEVC 4k bitstreams SHALL represent square pixels. Therefore, `aspect_ratio_idc` SHALL be set to 1.

The following restrictions SHALL apply for the fields in the sequence parameter set:

- `vui_parameters_present_flag = 1`
- `sps_extension_flag = 0`
- `fixed_pic_rate_general_flag = 1`
- `general_interlaced_source_flag = 0`

The following restrictions SHALL apply for the fields in the `profile_tier_level` syntax structure in the sequence parameter set:

- `general_tier_flag = 0`
- `general_profile_idc = 2`

UHD HEVC 4k bitstreams shall obey the limits in [MPEGHEVC] table A.1 and table A.2 associated to Level 5.1. `general_level_idc` shall be less than or equal to 153 (level 5.1).

Bitstreams which are compliant with the Main or Main10 profile SHOULD set `general_profile_compatibility_flag[1]` to 1.

The chromaticity coordinates of the ideal display, opto-electronic transfer characteristic of the source picture and matrix coefficients used in deriving luminance and chrominance signals from the red, green and blue primaries SHALL be explicitly signaled in the encoded HEVC Bitstream by setting the appropriate values for each of the following 3 parameters in the VUI: `colour_primaries`, `transfer_characteristics`, and `matrix_coeffs`.

[ITU-R-BT.709] colorimetry usage SHALL be signaled by setting `colour_primaries` to the value 1, `transfer_characteristics` to the value 1 and `matrix_coeffs` to the value 1.

The bitstream MAY contain SEI messages as permitted by [MPEGHEVC] and described in [MPEGHEVC] Annex D.

The `@codecs` parameter SHALL be set to either "hvc1.2.4.L153.B0" or "hev1.2.4.L153.B0" and SHALL NOT exceed the capabilities described by these values.

Bitstreams conforming to this chapter MAY contain one or more sets of optional dynamic metadata. The presence of dynamic metadata is signalled by a [supplemental property descriptor](#) with `@schemeIdUri="http://dashif.org/metadata/hdr"` and `@value` from the following table:

Scheme	@value
ETSI TS 103.433 SEI messages	TS103433

Figure 56 HEVC HDR dynamic metadata schemes.

11.17.1. TS 103.433 HDR dynamic metadata

This chapter applies to [video adaptation sets](#) that carry a [supplemental property descriptor](#) with `@schemeIdUri="http://dashif.org/metadata/hdr"` and `@value="TS103433"`.

The bitstream SHALL contain one or more SL-HDR Information SEI messages, as defined in clause A.2.2 of [ETSI103433-1], and MAY contain one or more Mastering Display Colour Volume SEI messages, as defined in [MPEGHEVC].

The SL-HDR Information SEI message SHALL be present at least with every SAP type 1 or type 2.

When carried, the Mastering Display Colour Volume SEI message SHALL be present at least with every SAP type 1 or type 2 and SHALL be used as specified in clause A.3 of [ETSI103433-1].

11.17.2. HEVC UHD compatibility aspects

This specification is designed such that UHD content that is authored in conformance to IOP is expected to conform to the media profile defined by [DVB-DASH] and following the 3GPP H.265/HEVC UHD Operation Point in section 5.6 of [3GPP26.116]. However, in contrast to DVB and 3GPP, only BT.709 may be used and not BT.2020.

In addition, clients conforming to this extension are expected to be capable of playing content authored as conform to the media profile defined by [DVB-DASH] and following the 3GPP H.265/HEVC UHD Operation Point in section 5.6 of [3GPP26.116], if BT.709 colour space is used.

11.18. HEVC HDR PQ10§

For the support of broad set of use cases addressing higher dynamic range (HDR) and wide colour gamut (WCG), the DASH-IF IOP HEVC HDR Perceptual Quantization (PQ) 10 Extension is defined. This interoperability point allows for additional UHD features including Wide Color Gamut, High Dynamic Range and a new electro-optical transfer curve. These features are in addition to the existing features described in the DASH-IF UHD 4k interoperability point, except that that this profile is designed for HDR, and requires the use of SMPTE ST 2084 [71] and Rec. BT-2020 [74] colour space. Note that this is identical to Rec. BT-2100 [80], PQ transfer function, Y'CB'R color difference formats, with 10 bit signal representation and narrow range.

Note that this Extension does not require the use of the maximum values, such as 60fps or 4K resolution. The content author may offer lower spatial and temporal resolutions and may use the regular DASH signalling to indicate the actual format of the source and rendering format. Typical cases may be to use HDR together with an HD 1080p signal. Note also that Adaptation Set Switching as defined in section 3.8 may be used to separate different spatial resolutions in different Adaptation Sets to address different capabilities, but still permit the use of lower resolutions for service continuity of higher resolutions.

The compliance to DASH-IF IOP HEVC HDR PQ10 may be signaled by a @profile attribute with the value <http://dashif.org/guidelines/dash-if-uhd#hevc-hdr-pq10>.

The same requirements as for UHD HEVC 4k as documented in section 10.2 hold, except for the changes as detailed below.

The changes in the HEVC HDR PQ10 profile that extend it beyond the HEVC 4K profile include:

- NAL Structured Video Streams conforming to this interoperability point SHALL be encoded using the REC-2020 color parameters as defined in [74]. Clients shall be able to correctly decode content that is encoded using that color space.
- NAL Structured Video Streams conforming to this interoperability point SHALL be encoded using the SMPTE ST 2084 electro-optic transfer function as defined in [71]. Clients shall be able to correctly decode content that is encoded using that electro-optic transfer function. Note that one cannot author a single piece of content that is compliant with both this profile and HEVC 4k profile. However, the content may be offered in one MPD in two different Adaptation Sets.

Optional metadata may be present in form SEI messages defined in ITU-T H.265 /ISO/IEC 230082:2015 [19].

A bitstream conforming to the HEVC HDR PQ10 media profile shall comply with the Main Tier Main10 Profile Level 5.1 restrictions, as specified in Recommendation ITU-T H.265 / ISO/IEC 23008-2 [19].

In addition the requirements in section 10.2.2.2 apply, except that this profile requires the use of Recommendation ITU-R BT.2020 [74] non-constant luminance colorimetry and SMPTE ST 2084 [71].

SMPTE ST 2084 [71] usage shall be signaled by setting colour primaries to the value 9, transfer_characteristics to the value 16 and matrix_coeffs to the value 9.

The bitstream may contain SEI messages as permitted by the Recommendation ITU-T H.265 / ISO/IEC 23008-2:2015 [19]. Details on these SEI messages are specified in Recommendation ITU-T H.265 / ISO/IEC 23008-2 / Annex D. SEI message may for example support adaptation of the decoded video signals to different display capabilities or more detailed content description, in particular those specified in Recommendation ITU-T H.265 / ISO/IEC 23008-2 / Annex D in relation to HDR. Other SEI Messages defined in ITU-T H.265 / ISO/IEC 23008-2 / Annex D may be present as well.

Receivers conforming to the HEVC HDR PQ10 media profile shall support decoding and displaying HEVC HDR PQ10 bitstreams as defined in section 10.3.2.2.

No additional processing requirements are defined, for example processing of SEI messages is out of scope.

If all Representations in an Adaptation Set conforms to the elementary stream constraints for the Media Profile as defined in clause 10.3.3.2 and the Adaptation Set conforms to the MPD signalling according to clause 10.3.3.2 and 10.3.3.4, and the Representations conform to the file format constraints in clause 10.3.3.3, then the @profiles parameter in the Adaptation Set may signal conformance to this operation point by using "<http://dashif.org/guidelines/dashif-uhd#hevc-hdr-pq10>".

The MPD shall conform to DASH-IF HEVC Main IOP as defined with the additional constraints defined in clause 10.3.3.4. The @codecs parameter shall not exceed and should be set to either "hvc1.2.4.L153.B0" or "hev1.2.4.L153.B0".

Content authored according to this extensions is expected to be interoperable with the HDR10 profile defined in the DECE CFF Content Specification v2.2 [78], although it should be noted that the DECE CFF profile may have additional constraints, such as bitrate restrictions and required metadata.

Content authored according to this extensions is expected to be interoperable with the PQ10 package defined in the UHD Forum Guidelines phase A [79].

11.18.1. HEVC PQ10 HDR dynamic metadata

Bitstreams conforming to the HEVC HDR PQ10 media profile may contain one or more sets of optional dynamic metadata. Details of the various metadata schemes are detailed below.

The presence of dynamic metadata is signalled by a Supplemental Descriptor with @schemeldUri set to "http://dashif.org/metadata/hdr", the @value set to one of the values in the following table:

Scheme	@value
SMPTE 2094-10 SEI messages	SMPTE2094-10
SMPTE 2094-40 SEI messages	SMPTE2094-40
TS 103.433 SEI messages	TS103433

Figure 57 HEVC HDR PQ10 dynamic metadata schemes

11.18.2. SMPTE 2094-10 HDR dynamic metadata

When the Adaptation Set contains a Supplemental Descriptor with @schemeldUri set to "http://dashif.org/metadata/hdr" and @value set to "SMPTE2094-10", then the bitstream shall contain SMPTE 2094-10 [83] metadata, provided as a Supplemental Enhancement Information (SEI) message containing a DM_data() message (as defined in SMPTE 2094-10 [83] Annex C- Display Management Message) in accordance with "User data registered by Recommendation ITU-T T.35 SEI message" syntax element.

In addition to the Bitstream Requirements defined above in Section 10.3.2.2, when ST2094-40 dynamic metadata is carried, exactly one ST 2094-10 SEI message shall be sent for every access unit of the bitstream.

11.18.3. SMPTE 2094-40 HDR dynamic metadata

When the Adaptation Set contains a Supplemental Descriptor with @schemeldUri set to "http://dashif.org/metadata/hdr" and @value set to "SMPTE2094-40", then the bitstream shall contain SMPTE ST 2094-40 [89] metadata, provided as a Supplemental Enhancement Information (SEI) message (as defined in CTA861-G [90]) in accordance with "User data registered by Recommendation ITU-T T.35 SEI message" syntax element.

This SEI message provides information to enable colour volume transformation of the reconstructed colour samples of the output pictures. The input to the indicated colour volume transform process is the linearized RGB colour components of the source content. The semantics and usage of the dynamic metadata shall be in conformance with the specifications in SMPTE ST 2094-40 [89].

In addition to the Bitstream Requirements defined above in clause 10.3.2.2, when ST2094-40 dynamic metadata is carried, exactly one ST 2094-40 SEI message shall be present with every SAP of type 1 or type 2.

11.19. UHD Dual-Stream (Dolby Vision)

[DolbyVision-ISOBMFF]

Note: This extension is designed to be compatible with the "Dolby Vision Media Profile Definition" in DECE "Common File Format & Media Formats Specification" Version 2.2. The name of the DASH-IF extension is inherited from the DECE document in order to indicate the compatibility with this DECE Media Profile.

For the support of broad set of backward compatible use cases the DASH-IF IOP Dual-Stream (Dolby Vision) Interoperability Point is defined. Backward Compatible refers to a simple method for one delivery format to satisfy both an HDR client and an SDR client. This Interoperability Point allows for two interlocked video streams, as described in the clause 10.4.2 below (restrictions to Enhancement Layers and Annex D 1.1). These two layers are known as the Base and Enhancement layers, where the Base Layer fully conforms to previous non-UHD or UHD

DASHIF Interoperability point. The EL provides additional information, which combined with the BL in a composition process produces a UHD output signal, including Wide Color Gamut and High Dynamic Range signal at the client.

The compliance to DASH-IF IOP Dual-Stream (Dolby Vision) may be signaled by a @profile attribute on the Enhancement Layer with the value <http://dashif.org/guidelines/dash-if-uhd#dvduallayer>

The dual-stream solution includes two video streams, known as the Base Layer and the Enhancement Layer. The high-level overview of the dual-stream process is shown in Figure 26 Overview of Dual-stream System.

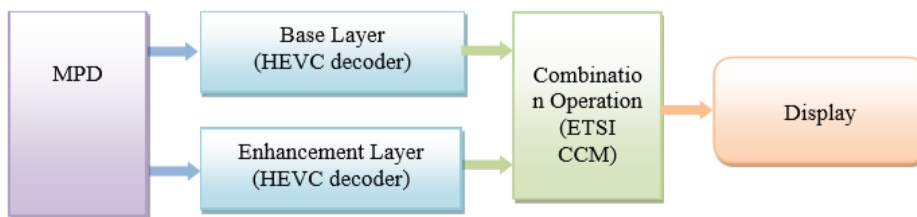


Figure 58 Overview of dual-stream system.

The MPD includes at least two Adaptation Sets as described below, including a Base Layer Adaptation Set and an Enhancement Layer Adaptation Set.

The Base Layer shall conform to the requirements of one of the following Interoperability Points: the DASH-IF IOP Main Interoperability Point, the DASH-IF IOP UHD 4k Interoperability point or the DASH-IF IOP UHD HDR10 Interoperability point. Any client that is able to play DASHIF IOP Main content, DASH-IF IOP UHD 4k content, or DASH-IF IOP UHD HDR10 content as appropriate will be able to play the content from the Base Layer track as determined by the client capabilities. To be clear, the Base Layer is 100% conforming, with no changes or additional information, to the profile definition. A client that plays content conforming to the Base Layer profile will be able to play the Base Layer content with no modification and no knowledge of the Enhancement Layer or and Dolby Vision specific information. See Annex E, Sample MPD, for an example dual-layer MPD.

In addition, The Enhancement Layer shall conform to H.265/HEVC Main10 Profile Main Tier as defined in Recommendation ITU-T H.265 / ISO/IEC 23008-2, Level 5.1 or lower The Enhancement Layer shall conform to the following additional requirements:

- The Frame Rate is identical to the Base Layer video track.
- The EL DPB (Decoded Picture Buffer) shall support the same number of maximum frames as the maximum number of frames supported by the BL's DPB.
- If the Base layer sample contains an IDR picture, the Enhancement Layer sample must have an IDR picture at the same presentation time.
- Fragment durations and Presentation times are identical to the Base Layer video track. To clarify, "Presentation times are identical" means that for each picture at one layer, there shall be a picture at the other layer with the same presentation time.
- Each Enhancement Layer track has one and only one associated Base Layer video track (i.e. tracks are paired 1:1).

The client - may either play the Base Layer alone, in which case it complies with the requirements of those interoperability points, or the client plays the Base Layer and Enhancement Layer together, decoding both layers and combining them to produce a 12 bit enhanced HDR signal which conforms to REC.2020 color parameters and SMPTE-2084 electro-optical transfer function. The details of this combination operation are detailed in ETSI Specification "Compound Content Management" [85].

Content shall only be authored claiming conformance to this IOP if a client can properly play the content through the method of combining the Base Layer and Enhancement layers to produce an enhanced HDR output. Note that clients who conform to the profile associated with the Base Layer alone may play the Base Layer alone, with no information (and no knowledge) of the Enhancement Layer. In addition, the content shall follow the mandatory aspects and should take into account the recommendations and guidelines for content authoring documented in sections 8 and 10 and HEVC-related issues in this section.

The dual-stream delivery of Dolby Vision asset uses two tracks; the Base Layer is written into one track according to the profile of the Base Layer, and the Enhancement Layer exists in a second track, per the [TBD Reference on integration, 12] specification and the details in Annex C and Annex D. In particular, details about required mp4 Boxes and sample entries are detailed in Annex C, "Dolby Vision Streams Within the ISO Base Media File Format" The Enhancement Layer is identified by an additional parameter, @dependencyId, which identifies the Base layer which is the match for the Enhancement Layer as described in clause 10.4.2.3.

If all Representations in an Adaptation Set conforms to the elementary stream constraints for the Media Profile as defined in clause 10.4.2.1 and the Adaptation Set conforms to the MPD signaling according to clause 10.4.3.2 and 10.4.3.3, and the Representations conform to the file format constraints in clause 10.4.3.4, then the @profiles parameter in the Adaptation Set may signal conformance to this operation point by using "http://dashif.org/guidelines/dash-ifuhd#dvduallayer on the Enhancement Layer (the Base Layer uses the normal signaling of the layer as defined in the profile of the Base Layer).

The MPD shall conform to DASH-IF HEVC Main IOP as defined with the additional constraints defined in clause 10.4.2.

When the Dual-Stream Dolby Vision asset is delivered as two files, the Enhancement Layer is identified by an additional parameter, @dependencyId, which identifies the Base Layer that is the match for the Enhancement Layer. The Base Layer Representation element must have an @id attribute, and the @dependencyId attribute on the Enhancement Layer Representation shall refer to that @id, to indicate to a client that these two representations are linked. Note that in this case, the @codecs attribute for the Base Layer will have only the Base Layer codec. In this example, the Base Layer @codecs might be: `codecs="hvc1.1.0.L120.00"` And the Enhancement Layer @codecs would be: `codecs="dvhe.dtr.uhd30"` For both the Base Layer and the Enhancement Layer, HEVC decoders are used in accordance with the @codecs signaling on each layer. The syntax and semantics of the @codecs signaling on the enhancement layer is detailed in Annex D. The output of the decoders are combined by the method detailed in ETSI Specification "Compound Content Management" [85].

Content shall only be authored claiming conformance to this IOP if 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 clause 8 and 10 and HEVC-related issues in clause 6.2.

11.19.1. Requirements for enhancement layer

The sample aspect ratio information shall be signaled in the bitstream using the aspect_ratio_idc value in the Video Usability Information (see values of aspect_ratio_idc in Recommendation ITU-T H.265 / ISO/IEC 23008-2:2013 [19], table E1).

In addition to the provisions set forth in Recommendation ITU-T H.265 / ISO/IEC 23008-2:2013 [19], the following restrictions shall apply for the fields in the sequence parameter set:

- bit_depth_luma_minus8 shall be set to "2".
- aspect_ratio_idc shall be set to "1".
- general_interlaced_source_flag shall be set to "0".

In addition to the requirements imposed in clause 10.4.2.2, the following additional specifications shall apply to the Enhancement Layer encoding: HEVC Enhancement Layer Bitstreams shall contain the following SEI messages:

- User data registered by Recommendation ITU-T T.35 SEI message containing the message CM_data() (named composing metadata SEI message), as described in clause 10.4.2.3.3.
- User data registered by Recommendation ITU-T T.35 SEI message containing the message DM_data() (named display management SEI Message), as described in clause 10.4.2.3.4.
- Mastering display colour volume SEI message as specified in Recommendation ITU-T H.265 / ISO/IEC 23008-2 Annex D with the following constraints: o A valid number shall be set for the following syntax elements: display primaries_x[c], display primaries_y[c], white_point_x, white_point_y, max_display_mastering_luminance and min_display_mastering_luminance.

CM_data() messages and DM_data() messages are carried in the enhancement layer video elementary stream as Supplemental Enhancement Information in HEVC's "User data registered by Recommendation ITU-T T.35 SEI message" syntactic element. The syntax of the composing metadata SEI message and the display management SEI message is defined in Table 31.

Field	Type	Usage
user_data_registered_itu_t_t35(payloadSize) {		
itu_t_t35_country_code	b(8)	This 8-bit field shall have the value 0xB5.
itu_t_t35_provider_code	u(16)	This 16-bit field shall have the value 0x0031.
user_identifier	u(32)	This 32-bit code shall have the value 0x47413934 ("GA94").
user_data_type_code	u(8)	An 8-bit value that identifies the type of user data to follow in the user_data_type_structure(). The values are defined in Table 32.
user_data_type_structure()		This is a variable length set of data defined by the value of user_data_type_code and table C.1 (DM_data()) or table D.1 (CM_data()).
}		

Figure 59 Compound Content Management SEI message: HEVC (prefix SEI NAL unit with nal_unit_type = 39, payloadType=4)

user_data_type_code	user_data_type_structure()
0x00 to 0x07	Reserved
0x08	CM_data()
0x09	DM_data()
0x0A to 0xFF	Reserved

Figure 60 User identifier

The composing metadata SEI message is a “user data registered by Recommendation ITU-T T.35 SEI message” containing a CM_data() message, as specified in Annex F. HEVC Enhancement Layer Bitstreams shall contain composing metadata SEI messages with the following constraints:

- It shall be sent for every access unit of the HEVC Enhancement Layer Bitstream.
- Bitstreams shall conform to ETSI Profile 1 as defined in Annex A of [85] and the value of the syntax element ccm_profile shall be set to “1”.
- The value of the syntax element ccm_level shall be set to “0”.
- The value of BL_bit_depth_minus8 shall be set to “2”.
- The value of EL_bit_depth_minus8 shall be set to “2”.
- The value of the syntax element hdr_bit_depth_minus8 shall be set to “2” or “4”.

The display management SEI message is a “user data registered by Recommendation ITU-T T.35 SEI message” containing a DM_data() message, as specified in Annex C. HEVC Enhancement Layer Bitstreams shall contain display management SEI messages with the following constraints:

- It shall be sent for every access unit of the HEVC Enhancement Layer Bitstream.
- app_identifier shall be set equal to “1”.
- app_version shall be set equal to “1”.
- The number of extension blocks with ext_block_level equal to “1” shall be constrained to be equal to “1”.
- The number of extension blocks with ext_block_level equal to “2” shall be constrained to be less than or equal to “16”.
- The number of extension blocks with ext_block_level equal to “5” shall be constrained to be equal to “0” or “1”.

11.20. VP9§

VP9 [86] is an alternative video codec which may be used for SD, HD, and UHD spatial resolutions, as well as HDR10 and HDR12 bit depths (HDR + WCG); and frame rates of 24fps and higher. This codec provides significant bandwidth savings at equivalent qualities with respect to AVC/H.264. While not meant to replace AVC and HEVC, DASH presentations may include additional VP9 representations for playback on clients which support it.

For the integration in the context of DASH, the following applies for VP9:

- The encapsulation of VP9 video data in ISO BMFF is defined in the VP Codec ISOBMFF Binding specification [87]. Clients shall support both sample entries containing `vp09` and `vpcc` boxes, i.e. inband storage for `VPCodecConfigurationBox` + `VPCodecConfigurationRecord`.
- For delivery to consumer devices, only VP9 profile 0 (4:2:0 chroma subsampling and 8bit pixel depth), and profile 1 (4.2.0 chroma subsampling and 10- or 12-bit pixel depths) shall be used.
- Stream Access Points shall coincide with the beginning of key frames (uncompressed header field `frame_type` = 0) as defined in the VP9 Bitstream Specification [86] section 7.2. Only type-1 SAPs are supported. Fragmentation and segmentation shall occur only at these points.
- Codec and codec configuration signaling in the MPD shall occur using the codec string defined in the VP Codec Binding Specification [87], DASH Application section.
- Encryption shall be signaled by the same mechanisms as defined in Common Encryption for ISO-BMFF Containers 3rd edition. Subsample encryption is required as per the VP Codec ISO Media File Format Binding spec [87].

For VP9 video streams, if the `@bitstreamSwitching` flag is set to true, then the following additional constraints shall apply:

- 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`').
 - 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 `@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.
- All representations within the Adaptation set shall have the same picture aspect ratio.
- All VP9 decoders are required to support dynamic video resolutions, however pixel bitdepths may not vary within an adaptation set. Because of this the encoding Profile must remain constant, but the Level may vary.
- All Representations within a video Adaptation Set shall include an Initialization Segment containing an '`vpcc`' Box containing a Decoder Configuration Record with the highest, , Level, vertical and horizontal resolutions of any Media Segment in the Representation.
- The `AdaptationSet@codecs` attribute shall be present and contain the maximum level of any Representation contained in the Adaptation Set.
- The `Representation@codecs` attribute may be present and in that case shall contain the maximum level of any Segment in the Representation.

The scope of the DASH-IF VP9-HD extension interoperability point is basic support of highquality video distribution over the top based on VP9 up to 1080p with 8-bit pixel depth and up to 30fps. Both, live and on-demand services are supported.

The compliance to DASH-VP9 main may be signaled by a `@profiles` attribute with the value "`http://dashif.org/guidelines/dashif#vp9`"

11.20.1. HD

A DASH client conforms to this extension IOP by supporting at least the following features:

- All DASH-related features as defined in clause 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.
- Content protection based on common encryption and key rotation as defined in section 7. And specifically, the client supports MPD-based parsing parameters for common encryption.
- All VP9 DASH IF IOP requirements in clause 11.2.
- VP9 Profile 0 up to level 4.1.

11.20.2. UHD§

The scope of the DASH-IF VP9-UHD extension interoperability point is basic support of highquality video distribution over the top based on VP9 up to 2160p with 8-bit pixel depth and up to 60fps. Both, live and on-demand services are supported. The compliance to DASH-VP9 main may be signaled by a @profiles attribute with the value "http://dashif.org/guidelines/dash-if-uhd#vp9"

A DASH client conforms to this extension IOP by supporting at least the following features:

- All features supported by DASH-IF VP9-HD defined in clause 11.3.1.
- VP9 Profile 0 up to level 5.1.

11.20.3. HDR§

The scope of the DASH-IF VP9-HDR extension interoperability point is basic support of highquality video distribution over the top based on VP9 up to 2160p with 10-bit pixel depth and up to 60fps. Both, live and on-demand services are supported.

The compliance to DASH-VP9 main may be signaled by a @profiles attribute with the value http://dashif.org/guidelines/dashif#vp9-hdr (up to HD/1080p resolution), or http://dashif.org/guidelines/dash-if-uhd#vp9-hdr (up to 4K resolution).

A DASH client conforms to this extension IOP by supporting at least the following features:

- All features supported by DASH-IF VP9-UHD defined in clauses 11.3.2.
- VP9 profile 2 up to level 5.1.
- Pixel depths of 10 bits.

12. Content protection and security§

12.1. Introduction§

DASH-IF do not intend to specify a full end-to-end DRM system. However DASH-IF provides a framework allowing multiple DRM systems to protect DASH content by adding private information in predetermined locations in MPDs and DASH content that is encrypted with Common Encryption as defined in [\[MPEGCENC\]](#).

Common Encryption specifies several protection schemes and associated parameters. These can be applied by a scrambling system and used by key mapping methods part of different DRM systems, thanks to common key identifiers (`KID` and `default_KID`). The same encrypted version of DASH content can be combined with different DRM systems private information allowing licenses and keys retrieval (Protection System Specific Header Box `pssh` in the ISOBMFF file and `ContentProtection` elements in the MPD. The DRM systems are identified by specific DRM systemID.

The recommendations in this document constrain the encryption parameters and use of the encryption metadata to specific use cases for VOD and live content with key rotation.

12.2. HTTPS and DASH§

Transport security in HTTP-based delivery may be achieved by using HTTP over TLS (HTTPS) as specified in [\[RFC 8446\]](#). HTTPS is a protocol for secure communication which is widely used on the Internet and also increasingly used for content streaming, mainly for protecting:

- The privacy of the exchanged data from eavesdropping by providing encryption of bidirectional communications between a client and a server, and
- The integrity of the exchanged data against forgery and tampering.

As an MPD carries links to media resources, web browsers follow the W3C recommendation [\[mixed-content\]](#). To ensure that HTTPS benefits are maintained once the MPD is delivered, it is recommended that if the MPD is delivered with HTTPS, then the media also be delivered with HTTPS.

DASH also explicitly permits the use of HTTPS as a URI scheme and hence, HTTP over TLS as a transport protocol. When using HTTPS in an MPD, one can for instance specify that all media segments are delivered over HTTPS, by declaring that all the `BaseURL`'s are HTTPS based, as follow:

```
<BaseURL>https://cdn1.example.com/</BaseURL>
<BaseURL>https://cdn2.example.com/</BaseURL>
```

One can also use HTTPS for retrieving other types of data carried with a MPD that are HTTP-URL based, such as, for example, DRM licenses specified within the `ContentProtection` element:

```
<ContentProtection
  schemeIdUri="urn:uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
  value="DRMNAME version"
  <drm:License>https://MoviesSP.example.com/protect?license=kljksdfiowek</drm:License>
</ContentProtection>
```

It is recommended that HTTPS be adopted for delivering DASH content. It should be noted nevertheless, that HTTPS does interfere with proxies that attempt to intercept, cache and/or modify content between the client and the TLS termination point within the CDN. Since the HTTPS traffic is opaque to these intermediate nodes, they can lose much of their intended functionality when faced with HTTPS traffic.

While using HTTPS in DASH provides good protection for data exchanged between DASH servers and clients, HTTPS only protects the transport link, but does not by itself provide an enforcement mechanism for access control and usage policies on the streamed content. HTTPS itself does not imply user authentication and content authorization (or access control). This is especially the case that HTTPS provides no protection to any streamed content cached in a local buffer at a client for playback. HTTPS does not replace a DRM.

12.3. Content Encryption§

DASH content SHALL be encrypted with either the cenc or the cbcs Common Encryption protection schemes. Full specification of these protection schemes is given in [\[MPEGCENC\]](#) sections 10.1 and 10.4 respectively.

Note: These are two non-interoperable encryption modes. DASH content encrypted with the cenc protection scheme cannot be decrypted by a device supporting only the cbcs protection scheme and vice versa.

DASH content associated to Representations contained in one Adaptation Set SHALL be encrypted with the same protection scheme, either cenc or cbcs.

Note: This is to facilitate seamless switching within Adaptation Sets, out of concern that some clients may not be able to switch between Representations seamlessly if the Representations are not all encrypted using the same algorithm.

DASH content represented by an MPD MAY be encrypted with different protection schemes in different Adaptation Sets. There MAY be unencrypted Period, hence clear to encrypted transitions and the opposite are possible.

Note: It is up to the client to select Adaptation Sets that it is able to process in each Period based on the MPD and the client's capabilities. A client may select Adaptation Sets that are all encrypted using the same protection scheme but this is not mandatory.

ISSUE 37 Clients in browsers must assume what the CDM support as there is no standardized API for probing the platform for knowing which Common Encryption protection scheme is supported. A bug is open on W3C EME and a pull request exists [here](#) for the ISOBMFF file format bytestream and a proposal is open for probing the platform on the encryption mode supported.

12.4. ISOBMFF Support for Common Encryption and DRM§

12.4.1. ISOBMFF Structure Overview§

ISOBMFF carries content protection information in different locations. The following shows the boxes hierarchy and composition for relevant boxes, when using common encryption:

- `moov/pssh` (zero or one per DRM system)

Protection System Specific Header box, see [\[MPEGCENC\]](#) section 8.1.1.

It contains license acquisition data and/or keys for each DRM system in a format that is proprietary. `pssh` boxes may be stored in Initialization Segment in the Movie Header box `moov` or in Media Segments in the Movie Fragment box `moof`.

- moof/traf/senc (one if encrypted)

Sample Encryption box, see [\[MPEGCENC\]](#) section 7.1.

It may store initialization vectors (IVs) and subsample encryption ranges. It is stored in each Track Fragment box of an encrypted track, and the stored parameters are accessed using the Sample Auxiliary Information Offset box (`saio`) and the Sample Auxiliary Information Size box (`saiz`).

- moof/traf/saio (one if encrypted)

Sample Auxiliary Information Offset box, see [\[MPEG4\]](#) section 8.7.9.

It contains the offset to the IVs and of the subsample encryption byte ranges.

- moof/traf/saiz (one if encrypted)

Sample Auxiliary Information Size box, see [\[MPEG4\]](#) section 8.7.8.

It contains the size of the IVs and of the subsample encryption byte ranges.

- moov/trak/mdia/minf/stbl/stsd/sinf/schm (one if encrypted)

Scheme Type box, see [\[MPEG4\]](#) section 8.12.5 and [\[MPEGCENC\]](#) section 4.

It contains the encryption scheme, identified by a 4-character code, `cenc` or `cbcs`. It is stored in the Protection Scheme Information box (`sinf` see [\[MPEG4\]](#) section 8.12.1 and [\[MPEGCENC\]](#) section 4) that signals that the track is encrypted.

- moov/trak/mdia/minf/stbl/stsd/sinf/schi/tenc (one if encrypted)

Track Encryption box, see [\[MPEGCENC\]](#) section 8.2.1.

It specifies encryption parameters and a KID named `default_KID` valid for the entire track. It is in the Initialization Segment. Any KID in Sample Group Description boxes (`sgpd`) override the `tenc` parameters (`default_KID` as well as `default_isProtected`).

For key rotation (see section [§ 12.8.1 Periodic Re-Authorization](#)), these additional boxes are used:

- moof/pssh (zero or one per DRM system)

Protection System Specific Header box, see [\[MPEGCENC\]](#) section 8.1.1.

It contains license acquisition data and/or keys for each DRM system in a format that is proprietary. `pssh` boxes may be stored in Initialization Segment in the Movie Header box `moov` or in Media Segments in the Movie Fragment box `moof`.

- moof/traf/sbgrp (one per sample group)

Sample to Group box, see [\[MPEG4\]](#) and [\[MPEGCENC\]](#) section 5.

With `sgpd` of type `seig` it is used to indicate the KID applied to each sample and allow changing KID over time (i.e. “key rotation”, see [\[MPEGDASH\]](#) section 8.9.4). The keys corresponding to the KIDs referenced by sample groups must be available when the samples in a Segment are ready for decryption. Those keys may be conveyed in that Segment in `pssh` boxes. A version 1 `pssh` box may be used to list all KIDs values to enable removal of duplicate boxes if a file is defragmented.

- moof/traf/sgpd ‘seig’ (sample group entry) (one per sample group)

Sample Group Description box, see [\[MPEG4\]](#) section 8.9.3 and [\[MPEGCENC\]](#).

When of type `seig`, it is used to indicate the KID applied to each sample and allow changing KID over time (i.e. “key rotation”, see [\[MPEGDASH\]](#) section 8.9.4).

12.4.2. ISOBMFF Content Protection Constraints

There SHALL be identical values of `default_KID` in the Track Encryption box (`tenc`) of all DASH content in Representation referenced by one Adaptation Set, when the Adaptation Set is protected by encryption. Different Adaptation Sets MAY have equal or different values of `default_KID`.

Note: In cases where, for example, SD and HD and UHD content in Representations are available in one Presentation, different license rights may be required for each quality level. In such case, separate Adaptation Sets should be created for each quality level, each with a different value of `default_KID`.

ISSUE 38 In this context, it is possible that the several quality levels are available under the same license right. Add text explaining why a shall is the way to do.

pssh boxes SHOULD NOT be present in Initialization Segments, and cenc:pssh elements in ContentProtection elements SHOULD be used instead. If pssh boxes are present in the Initialization Segment, each Initialization Segment within one Adaptation Set SHALL contain an equivalent pssh box for each DRM systemID, i.e. license acquisition from any Representation is sufficient to allow switching between Representations within the Adaptation Set without acquiring a new license.

Note: pssh boxes in Initialization Segments may result in playback failure when a license request is initiated each time an Initialization Segment is processed, such as the start of each protected Representation, each track selection, and each bitrate switch. This content requires DASH clients that can parse the pssh box contents to determine the duplicate license requests and block them.

ISSUE 39 This seems like an unlikely problem in real client implementations. Do we know of clients that actually exhibit the problematic behavior? Look at EME and define if this is still a problem. Take advantage of the meeting in May with W3C

Note: The duplication of the pssh information in the Initialization Segment may cause difficulties in playback with EME based clients, i.e. content will fail unless clients build complex DRM specific license handling.

12.5. DASH MPD Support for Common Encryption and DRM

12.5.1. MPD Structure Overview

The main DRM components in the MPD are the ContentProtection element (see [MPEGDASH] section 5.3.7.2 - table 9, section 5.8.5.2 and section 5.8.4.1) that contains the URI for signaling the use of Common Encryption or the use of a specific DRM and the cenc: namespace extension ([MPEGCENC] section 11.2). The MPD contains such information to help the client to determine if it can possibly play back content.

12.5.1.1. ContentProtection Element for the mp4protection Scheme

A ContentProtection element with the @schemeIdUri value "urn:mpeg:dash:mp4protection:2011" signals that content is encrypted with the scheme indicated in the @value, either cenc or cbcs for the Common Encryption schemes supported by this guidelines, as specified in [MPEGCENC]. It may be used to identify the KID values using the @cenc:default_KID attribute (see § 12.5.1.3 cenc: Namespace Extension), also present in the 'tenc' box. The values of this attribute are the KIDs expressed in UUID string notation.

This element 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.

```
<ContentProtection
  schemeIdUri="urn:mpeg:dash:mp4protection:2011"
  value="cenc"
  cenc:default_KID="34e5db32-8625-47cd-ba06-68fca0655a72"/>
```

12.5.1.2. ContentProtection Element for the UUID Scheme

A ContentProtection element with the @schemeIdUri value equal to a UUID value signals that content keys can be obtained through a DRM system identified by the UUID. The @schemeIdUri uses a UUID URN with the UUID string equal to the registered DRM systemID for a particular DRM system. This is specified in [MPEG4] section 5.8.5.2. An example is:

```
<ContentProtection
  schemeIdUri="urn:uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
  value="DRMNAME version"/>
```

12.5.1.3. cenc: Namespace Extension

[MPEGCENC] section 11.2 defines the namespace "urn:mpeg:cenc:2013", for which the usual namespace prefix is cenc.

A pssh box is defined by each DRM system for use with their registered DRM systemID, and the same box can be stored in the MPD within a ContentProtection element using an extension element in the cenc: namespace. Examples are provided in [MPEGCENC] section 11.2. A @cenc:default_KID can also be stored under the ContentProtection element using the same cenc: namespace.

Carrying @cenc:default_KID attribute and a cenc:pssh element is useful to allow key identification, license evaluation, and license retrieval before Initialization Segments carrying the same information become available. This allows clients to spread license requests and avoid simultaneous requests from all clients at the instant that an Initialization Segment containing the default_KID values becomes available. With @cenc:default_KID indicated in the mp4protection ContentProtection element on each Adaptation Set, clients can determine if the required content key is available, or which licenses the client needs to be obtained before the @availabilityStartTime of the Presentation based on the default_KID of each AdaptationSet element selected.

Carrying @cenc:default_KID attribute and a cenc:pssh element 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 the default_KID values becomes available. With @cenc:default_KID indicated in the mp4protection ContentProtection element on each Adaptation Set, clients can determine if that key and this Presentation is not available to the viewer (e.g. without a purchase or a 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.

12.5.2. MPD Content Protections Constraints

For an encrypted Adaptation Set, At least one ContentProtection element SHALL be present in the AdaptationSet element and apply to all contained Representations.

A ContentProtection element for the mp4Protection Scheme (@schemeIdUri equals to "urn:mpeg:dash:mp4protection:2011" and @value=cenc or cbcs) SHALL be present in the AdaptationSet element if DASH content represented by the contained Representations are encrypted. This allows clients to recognize the Adaptation Set is encrypted with a Common Encryption scheme without the need to understand any DRM system specific UUID element. This ContentProtection element SHALL contain the attribute @cenc:default_KID. The tenc box that specifies the encoded track encryption parameters shall be considered the definitive source of the default KID since it contains the default_KID field. The @cenc:default_KID attribute SHALL match the tenc default_KID value. This allows general-purpose clients to identify the default KID from the MPD using a standard location and format without the need to understand any DRM system specific information format.

The cenc:pssh element SHOULD be present in the ContentProtection element for each DRM system identified by a DRM system encoded as a UUID. 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 for license acquisition.

Note: A client 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 DRM systemID equal to the element's UUID value. This allows clients to acquire a license using only information in the MPD, prior to downloading Segments.

The @value attribute of the ContentProtection element for UUID Scheme SHOULD contain the DRM system name and version number in a human readable form.

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">
  <cenc:pssh>
    YmFzZTY0IGVudG9kZWQgY29udGVudHMgb2YgkXBzc2iS1GJveCB3aXRoaXMgU31zdGVtSUQ=
  </cenc:pssh>
</ContentProtection>
```

12.6. Mix ISOBMFF and MPD Content Protections Constraints

For a DRM system uniquely identified by its DRM systemID, in the case where the cenc:pssh element is present in

the MPD and the `pssh` box is present in the Initialization Segment, the `cenc:pssh` 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.

The DRM systems allowing to access to protected DASH content are signaled in the MPD and possibly also in the ISOBMFF file. In both cases, the DRM system is uniquely identified with a DRM systemID. A list of known identifiers can be found in the DASH identifier [repository](#).

If the default KID changes (this requires a new content key acquisition) and therefore the `@cenc:default_KID` value needs to be updated, it SHALL be at the beginning of a Period. A different Initialization Segment is then indicated with a different `default_KID` signaled in the `tenc` box.

Note: A file associated with a single content key 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 requesting the same license again; but it is possible that some the DRM systems 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).

12.7. Client Interactions with DRM Systems

A client interacts with one or more DRM systems during playback in order to control the decryption of content. The interaction is made through a DRM client that is responsible of enabling connection to a DRM server. Some of the most important interactions are:

- Determining the availability of content keys.
- Communicating with the DRM system to acquire content keys, most of the time through a license request.

In these interactions, the client and DRM system use the `default_KID` values as a mechanism to communicate information regarding the capability to decrypt DASH content described by Adaptation Sets. A DRM system MAY also make use of other keys in addition to the one signalled by the `default_KID` value (e.g. in key derivation or sample variant schemes) but this SHALL be transparent to the client.

When starting playback of Adaptation Sets, a client SHALL determine the required set of content keys based on the `default_KID` values.

Upon determining that one or more required content keys are not in its possession, the client SHOULD interact with the DRM system and request them. The client MAY also request content keys that are known to be usable. Clients SHALL request all required content keys signalled by the `default_KID` values. The client and/or DRM system MAY batch multiple requests (and the respective responses) into a single transaction (for example to reduce the chattiness of license acquisition traffic).

For efficient license delivery, it is recommended that clients:

- Request content keys on the initial processing of an MPD or ISOBMFF if `ContentProtection` elements or Initialization Segments are available with license acquisition information. This is intended to avoid a large number of simultaneous 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 simultaneous license requests during a live presentation at `Period@startTime`.

12.8. Additional Constraints for Specific Use Cases

12.8.1. Periodic Re-Authorization

This section explains different options and tradeoffs to enable change in content keys (a.k.a. key rotation) on a given piece of content.

Note: The main motivation is to enable access rights changes at program boundaries, not as a measure to increase security of content encryption. The term *Periodic re-authorization* is therefore used here instead of *key rotation*. Note that periodic re-authorization is also one of the ways to implement counting of active streams as this triggers a connection to a license server.

The following use cases are considered:

- Consumption models such as live content, PPV, PVR, VOD, SVOD, live to VOD, network DVR. This includes

cases where live content is converted into another consumption model for e.g. catch up TV.

- Regional blackout where client location may be taken into account to deny access to content in a geographical area.

The following requirements are considered:

- Ability to force a client to re-authorize to verify that it is still authorized for content consumption.
- Support seamless and uninterrupted playback when content keys are rotated by preventing storms of license requests from clients (these should be spread out temporally where possible to prevent spiking loads at isolated times), by allowing quick recovery (the system should be resilient if the server or many clients fail), and by providing to the clients visibility into the key rotation signaling.
- Support of hybrid broadcast/unicast networks in which client may operate in broadcast-only mode at least some of the time, e.g. where clients may not always be able to download licenses on demand through unicast.

This also should not require changes to DASH and the standard processing and validity of MPDs.

12.8.1.1. Periodic Re-Authorization Content Protections Constraints

Key rotation SHOULD not occur within individual segments. It is usually not necessary to rotate keys within individual segments. This is because segment durations are typically short in live streaming services (on the order of a few seconds), meaning that a segment boundary is usually never too far from the point where key rotation is otherwise desired to take effect.

When key hierarchy is used (see [§ 12.8.1.2 Implementation Options](#))

- Each Movie Fragment box (`moof`) box SHOULD contain one `pssh` box per DRM system. This `pssh` box SHALL contain sufficient information for obtaining content keys for this fragment when combined with information, for this DRM system, from either the `pssh` box obtained from the Initialization Segment or the `cenc:pssh` element from the MPD and the `KID` value associated with each sample from the `seig` Sample Group Description box and `sbgp` Sample to Group box that lists all the samples that use a given `KID` value.
- Constraints defined in [§ 12.4.2 ISOBMFF Content Protection Constraints](#) SHALL apply to the EMM/root license (one license is needed per Adaptation Set for each DRM system).

ISSUE 40 To be reviewed in light of CMAF and segment/chunk and low latency.

12.8.1.2. Implementation Options

This section describes recommended approaches for periodic re-authorization. They best cover the use cases and allow interoperable implementation.

Note: Other approaches are possible and may be considered by individual implementers. An example is explicit signaling using e.g. `esmg` messages, and a custom key rotation signal to indicate future KIDs.

Period: A `Period` element is used as the minimum content key duration interval. Content key is rotated at the period boundary. This is a simple implementation and has limitations in the flexibility:

- The existing signaling in the MPD does not allow for early warning of change of the content key and associated decryption context, hence seamless transition between periods is not ensured.
- The logic for the creation of the periods is decided by content creation not DRM systems, hence boundaries may not be suited properly and periods may be longer than the desired key interval.

Key Hierarchy: Each DRM system has its own key hierarchy. In general, the number of levels in the key hierarchy varies among DRM systems. For interoperability purposes, only two levels need to be considered:

- Licenses for managing the rights of a user: This can be issued once for enforcing some scope of accessing content, such as a channel or library of shows (existing and future). It is cryptographically bound to one DRM system and is associated with one user ID. It enables access to licenses that control the content keys associated with each show it authorizes. There are many names for this type of licenses. In conditional access systems, a data construct of this type is called an entitlement management message (EMM). In the PlayReady DRM system, a license of this type is called a “root license”. There is no agreement on a common terminology.
- Licenses for accessing the content: This is a license that contains content keys and can only be accessed by devices that have been authorized. While licenses for managing rights are most of the time unique per user, the licenses for accessing the content are not expected to be unique and are tied to the content and not a user,

therefore these may be delivered with content in a broadcast distribution. In addition doing so allows real time license acquisition, and do not require repeating client authentication, authorization, and rebuilding the security context with each content key change in order to enable continuous playback without interruption cause be key acquisition or license processing. In conditional access systems, a data construct of this type is called an entitlement control message (ECM). In the PlayReady DRM system, a license of this type is called a “leaf license”. There is no agreement on a common terminology.

When using key hierarchy, the `@cenc:default_KID` value in the `ContentProtection` element, which is also in the `tenc` box, is the ID of the key requested by the DRM client. These keys are delivered as part of acquisition of the rights for a user. The use of key hierarchy is optional and DRM system specific.

ISSUE 41 For key hierarchy, add a sentence explaining that mixing DRM systems is possible with system constraints.

12.8.2. Low Latency

Low latency content delivery requires that all components of the end-to-end systems are optimized for reaching that goal. DRM systems and the mechanisms used for granting access also need to be used in specific manners to minimize the impact on the latency. DRM systems are involved in the access to content in several manners:

- Device initialization
- Device authorization
- Content access granting

Each of these steps can have from an impact on latency ranging from low to high. The following describes possible optimizations for minimizing the latency.

12.8.2.1. Licenses Pre-Delivery

In a standard playback session, a client, after receiving the DASH MPD, checks the `@cenc:default_KID` value (either part of the `mp4protection` element or part of a DRM system element). If the client already has a content key associated to this `KID` value, it can safely assume that it is able to get access to content. If it does not have such content key, then a license request is triggered. This process is done every time a MPD is received (change of `Period`, change of Live service, notification of MPD change ...). It would therefore be better that the client always has all keys associated to `@cenc:default_KID` values. One mechanism is license pre-delivery. Predelivery can be performed in different occasions:

- When launching the application, the client needs to perform some initialization and refresh of data, it therefore connects to many servers for getting updated information. The license server SHOULD allow the client to receive licenses for accessing content the client is entitled to. Typically, for subscription services, all licenses for all Live services SHOULD be delivered during this initialization step. It is the DRM system client responsibility to properly store the received information.
- The DRM system SHOULD have a notification mechanism allowing to trigger a client or a set of clients to out-of-band initiate licenses request, so that it is possible to perform license updates in advance. This typically allows pre-delivery of licenses when a change will occur at a `Period` boundary and, in this case, this also allow avoiding all clients connecting at almost the same time to the license server if not triggered in advance randomly.
- In case a device needs nevertheless to retrieve a license, the DRM system MAY also batch responses into a single transaction allowing to provide additional licenses (as explained in Section [§ 12.7 Client Interactions with DRM Systems](#)) that can be used in the future.

12.8.2.2. Key Hierarchy and CMAF Chunked Content

When a DRM system uses key hierarchy for protecting content, it adds DRM information in both possibly the Initialization Segment and in the content (in the `moof` box). The information in the `moof` box can allow the DRM client to know which root key to use decrypt the leaf license or to identify the already decrypted content key from a local protected storage. Most of the processing and logic is DRM system-specific and involves DRM system defined encryption and signaling. It may also include additional steps such as evaluating leaf license usage rules. Key hierarchy is one technique for enabling key rotation and it is not required to rotate content key at high frequency, typically broadcast TV has content key cryptoperiods of 10 seconds to few minutes.

CMAF chunked Content introduces `moof` boxes at a high frequency as it appears within segments and not only at the

beginning of a segment. One can therefore expect to have several `moof` boxes every second. Adding signaling SHOULD be done only in the `moof` box of the first chunk in a segment.

ISSUE 42 To be completed. Look at encryption: Key available for license server “early” for been able to generate licenses (root or leaf licenses). Avoid the license server been on the critical path. Encourage license persistence in the client.

12.8.3. Use of W3C Clear Key with DASH

When using W3C Clear Key key system with DASH [\[encrypted-media\]](#), Clear Key related signaling is included in the MPD with a `ContentProtection` element that has the following format.

The Clear Key `ContentProtection` element attributes SHALL take the following values:

- The UUID `e2719d58-a985-b3c9-781a-b030af78d30e` is used for the `@schemeIdUri` attribute.
- The `@value` attribute is equal to the string “ClearKey1.0”

W3C also specifies the use of the DRM `systemID` “1077efec-c0b2-4d02-ace3-3c1e52e2fb4b” in [\[eme-initdata-cenc\]](#) section 4 to indicate that tracks are encrypted with Common Encryption [\[MPEGCENC\]](#), and list the `KID` of content keys used to encrypt the track in a version 1 `pssh` box with that DRM `systemID`. However, the presence of this Common `pssh` box does not indicate whether content keys are managed by DRM systems or Clear Key management specified in this section. Browsers are expected to provide decryption in the case where Clear Key management is used, and a DRM system where a DRM key management system is used. Therefore, clients SHALL NOT rely on the signalling of DRM `systemID` `1077efec-c0b2-4d02-ace3-3c1e52e2fb4b` as an indication that the Clear Key mechanism is to be used.

W3C specifies that in order to activate the Clear Key mechanism, the client must provide Clear Key initialization data to the browser. The Clear Key initialization data consists of a listing of the default KIDs required to decrypt the content.

The MPD SHOULD NOT contain Clear Key initialization data. Instead, clients SHALL construct Clear Key initialization data at runtime, based on the default KIDs signaled in the MPD using `ContentProtection` elements with the `urn:mpeg:dash:mp4protection:2011` scheme.

When requesting a Clear Key license to the license server, it is recommended to use a secure connection as described in Section [§ 12.2 HTTPS and DASH](#).

It should be noted that clients receiving content keys through the Clear Key key system may not have the same robustness that typical DRM clients are required to have. When the same content keys are distributed to DRM clients and to weakly-protected or unprotected clients, the weakly-protected or unprotected clients become a weak link in the system and limits the security of the overall system.

12.8.4. License Acquisition URL XML Element `Laur1`

One or more `Laur1` elements MAY be added under the `ContentProtection` element. This element specifies the URL for a license server allowing to receive a license. It has the optional attribute `@licenseType` that is a string that provides additional information that is DRM-specific.

The name space for the `Laur1` element is `http://dashif.org/guidelines/ContentProtection`. The namespace shortname is recommended to be “dashif”.

The XML schema for this element is:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://dashif.org/guidelines/ContentProtection"
  targetNamespace="http://dashif.org/guidelines/ContentProtection">
  <xs:element name="Laur1" type="Laur1Type"/>
  <xs:complexType name="Laur1Type">
    <xs:simpleContent>
      <xs:extension base="xs:anyURI">
        <xs:attribute name="licenseType" type="xs:string"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:schema>
```

12.8.4.1. ClearKey Example Using Laur1

An example of a Clear Key `ContentProtection` element using `Laur1` is as follows. One possible value of `@licenseType` is "EME-1.0" when the license served by the Clear Key license server is in the format defined in [\[encrypted-media\]](#).

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 DASH-MPD.xsd http://dashif.org/guidelines/ContentProtection laur1.xsd"
xmlns="urn:mpeg:dash:schema:mpd:2011"
xmlns:dashif="http://dashif.org/guidelines/ContentProtection"
type="static" profiles="urn:mpeg:dash:profile:mp2t-simple:2011" minBufferTime="PT1.45">
  <Period id="42" duration="PT6158S">
    <AdaptationSet mimeType="video/mp2t" codecs="avc1.4D401F,mp4a">
      <ContentProtection schemeIdUri="urn:uuid:1077efec-c0b2-4d02-ace3-3c1e52e2fb4b" value="ClearKey1.0">
        <dashif:Laur1>https://clearKeyServer.foocompany.com</dashif:Laur1>
        <dashif:Laur1 licenseType="EME-1.0">file://cache/licenseInfo.txt</dashif:Laur1>
      </ContentProtection>
    </AdaptationSet>
  </Period>
</MPD>
```

When used with a `@licenseType` equal to "EME-1.0":

- The GET request for the license includes in the body the JSON license request format defined in [\[encrypted-media\]](#) section 9.1.3. The license request MAY also include additional authentication elements such as access token, device or user ID.
- The response from the license server includes in the body the Clear Key license in the format defined in [\[encrypted-media\]](#) section 9.1.4 if the device is entitled to receive the Content Keys.

13. Annex B§

ISSUE 43 Merge Annex B from 4.3 to live services chapter (if not already duplicated).

14. Annex: Dolby Vision streams within ISO BMFF§

ISSUE 44 Where is this used? Why is it an annex? Consider restructuring to improve usefulness.

This Annex defines the structures for the storage of Dolby Vision video streams in a file format compliant with the ISO base media file format (ISO BMFF). Example file formats derived from the ISO BMFF include the Digital Entertainment Content Ecosystem (DECE) Common File Format (CFF) and Protected Interoperable File Format (PIFF). Note, that the file format defined here is intended to be potentially compliant with the DECE media specifications as appropriate.

The Dolby Vision decoder configuration record provides the configuration information that is required to initialize the Dolby Vision decoder.

The Dolby Vision Configuration Box contains the following information:

```
Box Type 'dvcC'

Container DolbyVisionHEVC SampleEntry( 'dvhe'), or DolbyVisionHVC1 SampleEntry( 'dvh1'), or

Mandatory Yes

Quantity Exactly One
```

The syntaxes of the Dolby Vision Configuration Box and decoder configuration record are described below.

```

align(8) class DOVIDecoderConfigurationRecord
{
    unsigned int (8) dv_version_major;
    unsigned int (8) dv_version_minor;
    unsigned int (7) dv_profile;
    unsigned int (6) dv_level;
    bit (1) dv_metadata_present_flag;
    bit (1) el_present_flag;
    bit (1) bl_present_flag;
    const unsigned int (32)[5] reserved = 0;
}

class DOVIConfigurationBox extends Box('dvcC')
{
    DOVIDecoderConfigurationRecord() DOVIConfig;
}

```

The semantics of the Dolby Vision decoder configuration record is described as follows.

`dv_version_major` - specifies the major version number of the Dolby Vision specification that the stream complies with. A stream compliant with this specification shall have the value 1.

`dv_version_minor` - specifies the minor version number of the Dolby Vision specification that the stream complies with. A stream compliant with this specification shall have the value 0.

`dv_profile` – specifies the Dolby Vision profile. Valid values are Profile IDs as defined in Table B.1 of Signaling Dolby Vision Profiles and Levels, Annex B.

`dv_level` – specifies the Dolby Vision level. Valid values are Level IDs as defined in Table B.2 of Signaling Dolby Vision Profiles and Levels, Annex B.

`dv_metadata_present_flag` – if 1 indicates that this track contains the supplemental enhancement information as defined in clause 10.4.2.2.

`el_present_flag` – if 1 indicates that this track contains the EL HEVC video substream.

`bl_present_flag` – if 1 indicates that this track contains the BL HEVC video substream.

Note: The settings for these semantic values are specified in Section A.7.1 Constraints on EL Track.

This section describes the Dolby Vision sample entries. It is used to describe tracks that contain substreams that cannot necessarily be decoded by HEVC compliant decoders.

The Dolby Vision sample entries contain the following information:

Box Type	'dvhe', 'dvh1'
Container	Sample Description Box ('stsd')
Mandatory	Yes
Quantity	One or more sample entries of the same type may be present

The syntax for the Dolby Vision sample entries are described below.

```

class DolbyVisionHEVCSESampleEntry() extends HEVCSESampleEntry('dvhe')
{
    DOVIConfigurationBox() config;
}

class DolbyVisionHVC1SampleEntry() extends HEVCSESampleEntry('dvh1')
{
    DOVIConfigurationBox() config;
}

```

A Dolby Vision HEVC sample entry shall contain a Dolby Vision Configuration Box as defined in C.2.2.

`config` - specifies the configuration information required to initialize the Dolby Vision decoder for a Dolby Vision EL track encoded in HEVC.

Compressorname in the base class VisualSampleEntry indicates the name of the compressor used, with the value “\013DOVI Coding” being recommended (013 is 11, the length of the string “DOVI coding” in bytes).

The brand ‘dby1’ SHOULD be used in the compatible_brands field to indicate that the file is compliant with all Dolby Vision UHD Extension as outlined in this document. The major_brand shall be set to the ISO-defined brand, e.g. ‘iso6’.

A Dolby Vision video stream can be encapsulated in a single file as a dual-track file containing separate BL and EL tracks. Each track has different sample descriptions.

For the visual sample entry box in an EL track a DolbyVisionHEVCVisualSampleEntry (dvhe) or DolbyVisionHVC1VisualSampleEntry (dvh1) SHALL be used.

The visual sample entries SHALL contain an HEVC Configuration Box (hvcC) and a Dolby Vision Configuration Box (dvcc).

The EL track shall meet the following constraints:

- In the handler reference box, the handler_type field shall be set to ‘vide’.
- The media information header box shall contain a video media header box.
- The dependency between the Dolby Vision base and enhancement track shall be signaled by the tref box in the enhancement track.. The reference_type shall be set to vdep.
- The dv_profile field in the Dolby Vision Configuration Box (dvcc) shall be set according the encoded Dolby Vision profile.
- The dv_level field in the Dolby Vision Configuration Box (dvcc) shall be set according the encoded Dolby Vision level.
- The dv_metadata_present_flag shall be set to 1.
- The el_present_flag shall be set to 0 or 1.
- The bl_present_flag shall be set to 0.

The following table shows the box hierarchy of the EL track.

Nesting Level				Reference
4	5	6	7	
stbl				ISO/IEC 14496-12
	stsd			
		dvhe, or dvh1		Section A.3
			hvcC	
			dvcc	Section 3.1
	stts			ISO/IEC 14496-12
	stsc			
	stsz			
	stz2			
	stco			
	co64			

Figure 61 Sample table box hierarchy for the EL track of a dual-track Dolby Vision file

Note: This is not an exhaustive list of boxes.

For a dual-track file, the movie fragments carrying the BL and EL shall meet the following constraints:

- The adjacent movie fragments (moof and mdat) for the base and enhancement track shall be interleaved with BL followed by EL. BL and EL samples shall be placed in separate Movie Fragments and that each BL Movie Fragment shall be immediately followed by an EL movie fragment containing the same number of samples with identical composition timestamps.

The track fragment random access box (tfrac) for the base and enhancement track shall conform to the ISO/IEC 14496-12 (section 8.8.10) and meet the following additional constraint:

- The value of the time field in the track fragment random access box indicates the presentation time of a random accessible sample. This time value shall be identical for every corresponding random accessible sample in the base and enhancement track.

15. Annex: Signaling Dolby Vision profiles and levels

ISSUE 45 Where is this used? Why is it an annex? Consider restructuring to improve usefulness.

This Annex defines the detailed list of Dolby Vision profile/levels and how to represent them in a string format. This string can be used for identifying Dolby Vision device capabilities and identifying the type of the Dolby Vision streams presented to device through various delivery mechanisms such as HTML 5.0 and MPEG-DASH.

The Dolby Vision codec provides a rich feature set to support various ecosystems such as Over the Top streaming, Broadcast television, Blu-Ray discs, and OTT streaming. The codec also supports many different device implementation types such as GPU accelerated software implementation, full-fledged hardware implementation, and hardware plus software combination. One of the Dolby Vision codec features allows choosing the type of backward compatibility such as non-backward compatible or backward compatible with SDR. A Dolby Vision capable device may not have all the features or options implemented, hence it is critical the device advertises the capabilities and content server provides accurate Dolby vision stream type information.

Following are the currently supported Dolby Vision profiles:

Profile ID	Profile Name	BL Codec	EL Codec	BL:EL	BL Backward Compatibility*	BL/EL Full Alignment**	BL Codec Profile	EL Codec Profile
2	dvhe.der	HEVC8	HEVC8	1:1/4	SDR	No	H.265 Main	H.265 Main
3	dvhe.den	HEVC8	HEVC8	1:1	None	No	H.265 Main	H.265 Main
4	dvhe.dtr	HEVC10	HEVC10	1:1/4	SDR	No	H.265 Main10	H.265 Main10
5	dvhe.stn	HEVC10	N/A	N/A	None	N/A	H.265 Main10	N/A
6	dvhe.dth	HEVC10	HEVC10	1:1/4	HDR10	No	H.265 Main10	H.265 Main10
7	dvhe.dtb	HEVC10	HEVC10	1:1/4 for UHD	Blu-ray HDR	No	H.265 Main10	H.265 Main10
				1:1 for FHD				

Figure 62 Dolby Vision profiles

Legend:

BL:EL

ratio of Base Layer resolution to Enhancement Layer resolution (when applicable)

BL/EL Full alignment

The Enhancement Layer (EL) GOP and Sub-GOP structures are fully aligned with Base Layer (BL), i.e. the BL/EL IDRs are aligned, BL/EL frames are fully aligned in decode order such that skipping or seeking is possible anywhere in the stream not only limited to IDR. BL AU and EL AU belonging to the same picture shall have the same POC (picture order count)

Encoder Recommendations

- Dolby Vision Encoders should only use baseline profile composer for profiles which are non-backward compatible, i.e. the BL Backward Compatibility = None.
- Encoders producing Dolby Vision dual layer streams should generate BL/EL with full GOP/Sub-GOP structure alignment for all the profiles listed in Table 4.

The following is the profile string naming convention: `dv[BL codec type].[number of layers][bit depth][backward compatibility] [EL codec type][EL codec bit depth]`

Attribute	Syntax
dv	dv = Dolby Vision
BL codec type	he = HEVC
Number of layers	s = single layer d = dual layer without enforcement of BL/EL GOP structure and POC alignment p = dual layer with enforcement of BL/EL GOP structure and POC alignment
Bit depth	e = 8 t = 10
Backward compatibility	n = non-backward compatible r = SDR backward compatible (rec.709, 100 nits) h = HDR10 backward compatible b = Blu-ray backward compatible (Ultra HD Blu-ray™ High Dynamic Range)
EL codec Type (see Note 1 below)	a = AVC h = HEVC
EL codec bit depth (See Note 1 below)	e = 8 t = 10

Figure 63 Components of a Dolby Vision @codecs string.

Notes:

1. [EL codec type] and [EL codec bit depth] shall only be present if the EL codec type is different from the BL codec.
2. Interlaced: There is no support for interlaced video at this time.
3. Codecs other than HEVC or AVC may be supported in future.

The Dolby Vision level indicates the maximum frame rate and resolution supported by the device for a given profile. Typically there is a limit on the maximum number of pixels the device can process per second in a given profile; the level indicates the maximum pixels and the maximum bitrate supported in that profile. Since maximum pixels per second is a constant for given level, the resolution can be reduced to get higher frame rate and vice versa. Following are the possible levels:

Level ID	Level Name	Example Max Resolution x FPS	Max Bit Rates (BL and EL combined)			
			main tier (Mbps)	high tier (Mbps)	main tier (Mbps)	high tier (Mbps)
1	hd24	1280x720x24	20	50		
2	hd30	1280x720x30	20	50		
3	fhd24	1920x1080x24	20	70		
4	fhd30	1920x1080x30	20	70		
5	fhd60	1920x1080x60	20	70		
6	uhd24	3840x2160x24	25	130		
7	uhd30	3840x2160x30	25	130		
8	uhd48	3840x2160x48	40	130		
9	uhd60	3840x2160x60	40	130		

Figure 64 Dolby Vision levels.

The following is the level string naming convention: [resolution][fps][high tier]

Attribute	Syntax
Resolution	hd = 720 fhd = 1080 uhd = 2160
FPS	Frames per second (e.g. 24, 30, 60)
High Tier	Whether or not higher frame rates are supported. If yes, “h” will be appended

Figure 65 Components of a Dolby Vision level string.

The profile and level string is recommended to be joined in the following manner: Format: [Profile String].[Level String]

Examples

dvav.per.fhd30

dual layer avc 8 bit with enforcement of BL/EL GOP Structure and POC alignment, rec709 backwards compatible, 1920x1080@30fps

dvhe.stn.uhd30

single layer hevc 10 bit non-backwards compatible, 3840x2160@30fps

The device capabilities can be expressed in many ways depending on the protocol used by the streaming service or VOD service. The device could maintain a list of supported capabilities in an array:

```
String capabilities [] = {"dvhe.dtr.uhd24", "dvhe.stn.uhd30"}
```

After receiving the manifest the Player could iterate over the stream types and check whether a stream type is supported by searching the capabilities[].

When using HTTP, the device could send the capabilities via the user agent string in HTTP request in following manner:

```
Opera/9.80 (Linux armv71) Presto/2.12.407 Version/12.51 Model-UHD+dvhe.dtr.uhd24+dvhe.stn.uhd30/1.0.0 (Manufacturer name, Model)
```

A server program can search for +dv to determine whether Dolby Vision is supported and further identify the profiles and level supported by parsing the characters following the +dv. Multiple profiles/level pairs can be listed with + beginning each profile/level pair.

16. Annex: Display management message§

ISSUE 46 Where is this used? Why is it an annex? Consider restructuring to improve usefulness.

A display management (DM) message contains metadata in order to provide dynamic information about the colour volume of the video signal. This metadata can be employed by the display to adapt the delivered HDR imagery to the capability of the display device. The information conveyed in this message is intended to be adequate for purposes corresponding to the use of Society of Motion Picture and Television Engineers ST 2094-1 and ST 2094-10.

The syntax and semantics for `DM_data()` are defined in clause C.2.

<code>DM_data () {</code>	Descriptor
app_identifier	ue(v)
app_version	ue(v)
metadata_refresh_flag	u(1)
if(metadata_refresh_flag) {	
num_ext_blocks	ue(v)
if(num_ext_blocks) {	
while(!byte_aligned())	
dm_alignment_zero_bit	f(1)
for(i = 0; i < num_ext_blocks; i ++) {	
ext_dm_data_block(i)	
}	
}	
}	
while(!byte_aligned())	
dm_alignment_zero_bit	f(1)
}	

Figure 66 `DM_data()`

<code>ext_dm_data_block() {</code>	Descriptor
ext_block_length	ue(v)
ext_block_level	u(8)
ext_dm_data_block_payload(ext_block_length, ext_block_level)	
}	

Figure 67 `ext_dm_data_block()`

ext_dm_data_block_payload(ext_block_length, ext_block_level Descriptor) {	
ext_block_len_bits = 8 * ext_block_length	
ext_block_use_bits = 0	
if(ext_block_level == 1) {	
min_PQ	u(12)
max_PQ	u(12)
avg_PQ	u(12)
ext_block_use_bits += 36	
}	
if(ext_block_level == 2) {	
target_max_PQ	u(12)
trim_slope	u(12)
trim_offset	u(12)
trim_power	u(12)
trim_chroma_weight	u(12)
trim_saturation_gain	u(12)
ms_weight	i(13)
ext_block_use_bits += 85	
}	
if(ext_block_level == 5) {	
active_area_left_offset	u(13)
active_area_right_offset	u(13)
active_area_top_offset	u(13)
active_area_bottom_offset	u(13)
ext_block_use_bits += 52	
}	
while(ext_block_use_bits++ < ext_block_len_bits)	
ext_dm_alignment_zero_bit	f(1)
}	

Figure 68 ext_dm_data_block_payload()

This clause defines the semantics for DM_data().

For the purposes of the present clause, the following mathematical functions apply:

$$\text{Abs}(x) = \begin{cases} x & ; x \geq 0 \\ -x & ; x < 0 \end{cases}$$

Floor(x) is the largest integer less than or equal to x.

$$\text{Sign}(x) = \begin{cases} 1 & ; x > 0 \\ 0 & ; x = 0 \\ -1 & ; x < 0 \end{cases}$$

$$\text{Clip3}(x, y, z) = \begin{cases} x & ; z < x \\ y & ; z > y \\ z & ; \text{otherwise} \end{cases}$$

$$\text{Round}(x) = \text{Sign}(x) * \text{Floor}(\text{Abs}(x) + 0.5)$$

/ = Integer division with truncation of the result toward zero. For example, 7/4 and -7/-4 are truncated to 1 and -1 respectively. -7/4 and 7/-4 are truncated to -1.

app_identifier

identifies an application in the ST 2094 suite.

app_version

specifies the application version in the application in the ST 2094 suite.

metadata_refresh_flag

when set equal to 1 cancels the persistence of any previous extended display mapping metadata in output order and indicates that extended display mapping metadata follows. The extended display mapping metadata persists from the coded picture to which the SEI message containing DM_data() is associated (inclusive) to the coded picture to which the next SEI message containing DM_data() and with metadata_refresh_flag set equal to 1 in output order is associated (exclusive) or (otherwise) to the last picture in the coded video sequence (inclusive). When set equal to 0 this flag indicates that the extended display mapping metadata does not follow.

num_ext_blocks

specifies the number of extended display mapping metadata blocks. The value shall be in the range of 1 to 254, inclusive.

dm_alignment_zero_bit

shall be equal to 0

ext_block_length[i]

is used to derive the size of the i-th extended display mapping metadata block payload in bytes. The value shall be in the range of 0 to 1023, inclusive.

ext_block_level[i]

specifies the level of payload contained in the i-th extended display mapping metadata block. The value shall be in the range of 0 to 255, inclusive. The corresponding extended display mapping metadata block types are defined in Table E.1.4. Values of ext_block_level[i] that are ATSC reserved shall not be present in the bitstreams conforming to this version of ATSC specification. Blocks using ATSC reserved values shall be ignored.

When the value of ext_block_level[i] is set equal to 1, the value of ext_block_length[i] shall be set equal to 5.

When the value of ext_block_level[i] is set equal to 2, the value of ext_block_length[i] shall be set equal to 11.

When the value of ext_block_level[i] is set equal to 5, the value of ext_block_length[i] shall be set equal to 7.

ext_block_level	extended metadata block type
0	Reserved
1	Level 1 Metadata – Content Range
2	Level 2 Metadata – Trim Pass
3	Reserved
4	Reserved
5	Level 5 Metadata – Active Area
6...255	Reserved

Figure 69 Definition of extended display mapping metadata block type.

When an extended display mapping metadata block with ext_block_level equal to 5 is present, the following constraints shall apply:

- An extended display mapping metadata block with `ext_block_level` equal to 5 shall be preceded by at least one extended display mapping metadata block with `ext_block_level` equal to 1 or 2.
- Between any two extended display mapping metadata blocks with `ext_block_level` equal to 5, there shall be at least one extended display mapping metadata block with `ext_block_level` equal to 1 or 2.
- No extended display mapping metadata block with `ext_block_level` equal to 1 or 2 shall be present after the last extended display mapping metadata block with `ext_block_level` equal to 5
- The metadata of an extended display mapping metadata block with `ext_block_level` equal to 1 or 2 shall be applied to the active area specified by the first extended display mapping metadata block with `ext_block_level` equal to 5 following this block.

When the active area defined by the current extended display mapping metadata block with `ext_block_level` equal to 5 overlaps with the active area defined by preceding extended display mapping metadata blocks with `ext_block_level` equal to 5, all metadata of the extended display mapping metadata blocks with `ext_block_level` equal to 1 or 2 associated with the current extended display mapping metadata block with `ext_block_level` equal to 5 shall be applied to the pixel values of the overlapping area.

min_PQ specifies the minimum luminance value of the current picture in 12-bit PQ encoding. The value shall be in the range of 0 to 4095, inclusive. Note that the 12-bit `min_PQ` value with full range is calculated as follows:

```
min_PQ = Clip3(0, 4095, Round(Min * 4095))
```

where `Min` is `MinimumPqencodedMaxrgb` as defined in clause 6.1.3 of SMPTE ST 2094-10.

max_PQ specifies the maximum luminance value of current picture in 12-bit PQ encoding. The value shall be in the range of 0 to 4095, inclusive. Note that the 12-bit `max_PQ` value with full range is calculated as follows:

```
max_PQ = Clip3(0, 4095, Round(Max * 4095))
```

where `Max` is `MaximumPqencodedMaxrgb` as defined in clause 6.1.5 of SMPTE ST 2094-10.

avg_PQ specifies the midpoint luminance value of current picture in 12-bit PQ encoding. The value shall be in the range of 0 to 4095, inclusive. Note that the 12-bit `avg_PQ` value with full range is calculated as follows:

```
avg_PQ = Clip3(0, 4095, Round(Avg * 4095))
```

where `Avg` is `AveragePqencodedMaxrgb` as defined in section 6.1.4 of SMPTE ST 2094-10.

target_max_PQ specifies the maximum luminance value of a target display in 12-bit PQ encoding. The value shall be in the range of 0 to 4095, inclusive. The `target_max_PQ` is the PQ encoded value of `TargetedSystemDisplayMaximumLuminance` as defined in clause 10.4 of SMPTE ST 2094-1.

If there is more than one extended display mapping metadata block with `ext_block_level` equal to 2, those blocks shall have no duplicated `target_max_PQ`.

trim_slope specifies the slope metadata. The value shall be in the range of 0 to 4095, inclusive. If `trim_slope` is not present, it shall be inferred to be 2048. Note that the 12-bit slope value is calculated as follows:

```
S_S = Clip3(0, 4095, Round((S-0.5) * 4096))
```

where `S` is the `ToneMappingGain` as defined in clause 6.2.3 of SMPTE ST 2094-10.

trim_offset specifies the offset metadata. The value shall be in the range of 0 to 4095, inclusive. If `trim_offset` is not present, it shall be inferred to be 2048. Note that the 12-bit offset value is calculated as follows:

```
S_O = Clip3(0, 4095, Round((S+0.5) * 4096))
```

where `O` is the `ToneMappingOffset` as defined in clause 6.2.2 of SMPTE ST 2094-10.

trim_power specifies the power metadata. The value shall be in the range of 0 to 4095, inclusive. If `trim_power` is not present, it shall be inferred to be 2048. Note that the 12-bit power value is calculated as follows:

```
P_P = Clip3(0, 4095, Round((P-0.5) * 4096))
```

where `P` is the `ToneMappingGamma` as defined in clause 6.2.4 of SMPTE ST 2094-10.

trim_chroma_weight specifies the chroma weight metadata. The value shall be in the range of 0 to 4095, inclusive. If **trim_chroma_weight** is not present, it shall be inferred to be 2048. Note that the 12-bit chroma weight value is calculated as follows:

$$\text{trim_chroma_weight} = \text{Clip3}(0, 4095, \text{Round}((\text{CW} + 0.5) * 4096))$$

where CW is the ChromaCompensationWeight as defined in clause 6.3.1 of SMPTE ST 2094-10.

trim_saturation_gain specifies the saturation gain metadata. The value shall be in the range of 0 to 4095, inclusive. If **trim_saturation_gain** is not present, it shall be inferred to be 2048. Note that the 12-bit saturation gain value is calculated as follows:

$$\text{trim_saturation_gain} = \text{Clip3}(0, 4095, \text{Round}((\text{SG} + 0.5) * 4096))$$

where SG is the SaturationGain as defined in clause 6.3.2 of SMPTE ST 2094-10.

ms_weight specifies the multiscale weight metadata. The value shall be in the range of -1 to 4095, inclusive. If **ms_weight** is not present, it shall be inferred to be 2048. Where **ms_weight** is equal to -1, the bit stream indicates **ms_weight** is unspecified. The 13-bit multiscale weight value is calculated as follows:

$$\text{ms_weight} = -1 \text{ OR } \text{Clip3}(0, 4095, \text{Round}(\text{MS} * 4096))$$

where MS is the ToneDetailFactor as defined in clause 6.4.2 of SMPTE ST 2094-10.

active_area_left_offset, **active_area_right_offset**, **active_area_top_offset**, **active_area_bottom_offset** specify the active area of current picture, in terms of a rectangular region specified in picture coordinates for active area. The values shall be in the range of 0 to 8191, inclusive. See also UpperLeftCorner and LowerRightCorner definitions in ST 2094-1.

If **active_area_left_offset**, **active_area_right_offset**, **active_area_top_offset**, **active_area_bottom_offset** are not present, they shall be inferred to be 0.

The coordinates of top left active pixel is derived as follows:

$$X_{\text{top_left}} = \text{active_area_left_offset}$$

$$Y_{\text{top_left}} = \text{active_area_top_offset}$$

The coordinates of top left active pixel are defined as the UpperLeftCorner in clause 9.2 of SMPTE ST.2094-1.

With Xsize is the horizontal resolution of the current picture and Ysize is the vertical resolution of current picture, the coordinates of bottom right active pixel are derived as follows:

$$X_{\text{bottom_right}} = \text{XSize} - 1 - \text{active_area_right_offset}$$

$$Y_{\text{bottom_right}} = \text{YSize} - 1 - \text{active_area_bottom_offset}$$

where $X_{\text{bottom_right}}$ greater than $X_{\text{top_left}}$ and $Y_{\text{bottom_right}}$ greater than $Y_{\text{top_left}}$.

The coordinates of bottom right active pixel are defined as the LowerRightCorner in clause 9.3 of SMPTE ST.2094-1.

ext_dm_alignment_zero_bit shall be equal to 0.

17. Annex: Composing metadata message§

ISSUE 47 Where is this used? Why is it an annex? Consider restructuring to improve usefulness.

A composing metadata (CM) message contains the metadata which is needed to apply the post-processing process as described in the ETSI ETCCM specification to recreate the HDR UHDTV pictures.

The syntax for CM_data() is shown in table D.1. The number of bits “v” used to represent each of the syntax elements of CM_data(), for which the parsing process is specified by the descriptor u(v), is defined in table D.2.

CM_data() {	Descriptor
ccm_profile	u(4)
ccm_level	u(4)
coefficient_log2_denom	ue(v)
BL_bit_depth_minus8	ue(v)
EL_bit_depth_minus8	ue(v)
hdr_bit_depth_minus8	ue(v)
disable_residual_flag	u(1)
for(cmp = 0; cmp < 3; cmp++) {	
num_pivots_minus2 [cmp]	ue(v)
for (pivot_idx = 0; pivot_idx < num_pivots_minus2[cmp] + 2; pivot_idx ++) {	
pred_pivot_value [cmp][pivot_idx]	u(v)
} // end of pivot points for BL three components	
} //cmp	
for (cmp = 0; cmp < 3; cmp++) { //mapping parameters	
for (pivot_idx = 0; pivot_idx < num_pivots_minus2[cmp] + 1; pivot_idx++) {	
mapping_idc [cmp][pivot_idx]	ue(v)
if(mapping_idc [cmp][pivot_idx] == MAPPING_POLYNOMIAL) {	
poly_order_minus1 [cmp][pivot_idx]	ue(v)
for(i = 0; i <= poly_order_minus1[cmp][pivot_idx] + 1; i ++) {	
poly_coef_int [cmp][pivot_idx][i]	se(v)
poly_coef [cmp][pivot_idx][i]	u(v)
}	
else if(mapping_idc [cmp][pivot_idx] == MAPPING_MMR) {	
mmr_order_minus1 [cmp][pivot_idx]	u(2)
mmr_constant_int [cmp][pivot_idx]	se(v)
mmr_constant [cmp][pivot_idx]	u(v)
for(i = 1; i <= mmr_order_minus1 + 1; i ++) {	
for (j = 0; j < 7; j++) {	
mmr_coef_int [cmp][pivot_idx][i][j]	se(v)
mmr_coef [cmp][pivot_idx][i][j]	u(v)
} // the j-th coefficients	
} // the i-th order	
} // MMR coefficients	
} // pivot_idx	
} // cmp	
if (!disable_residual_flag) {	
for (cmp = 0; cmp < 3; cmp++) { //quantization parameters	
nlq_offset [cmp]	u(v)
hdr_in_max_int [cmp]	ue(v)
hdr_in_max [cmp]	u(v)
linear_deadzone_slope_int [cmp]	ue(v)
linear_deadzone_slope [cmp]	u(v)
linear_deadzone_threshold_int [cmp]	ue(v)
linear_deadzone_threshold [cmp]	u(v)
} // cmp	

Figure 70 CM_data()

CM_data() {	Descriptor
ccm_profile	u(4)
ccm_level	u(4)
coefficient_log2_denom	ue(v)
BL_bit_depth_minus8	ue(v)
EL_bit_depth_minus8	ue(v)
hdr_bit_depth_minus8	ue(v)
disable_residual_flag	u(1)
for(cmp = 0; cmp < 3; cmp++) {	
num_pivots_minus2 [cmp]	ue(v)
for (pivot_idx = 0; pivot_idx < num_pivots_minus2[cmp] + 2; pivot_idx ++) {	
pred_pivot_value [cmp][pivot_idx]	u(v)
} // end of pivot points for BL three components	
} //cmp	
for (cmp = 0; cmp < 3; cmp++) { //mapping parameters	
for (pivot_idx = 0; pivot_idx < num_pivots_minus2[cmp] + 1; pivot_idx++) {	
mapping_idc [cmp][pivot_idx]	ue(v)
if(mapping_idc [cmp][pivot_idx] == MAPPING_POLYNOMIAL) {	
poly_order_minus1 [cmp][pivot_idx]	ue(v)
for(i = 0; i <= poly_order_minus1[cmp][pivot_idx] + 1; i ++) {	
poly_coef_int [cmp][pivot_idx][i]	se(v)
poly_coef [cmp][pivot_idx][i]	u(v)
}	
else if(mapping_idc [cmp][pivot_idx] == MAPPING_MMR) {	
mmr_order_minus1 [cmp][pivot_idx]	u(2)
mmr_constant_int [cmp][pivot_idx]	se(v)
mmr_constant [cmp][pivot_idx]	u(v)
for(i = 1; i <= mmr_order_minus1 + 1; i ++) {	
for (j = 0; j < 7; j++) {	
mmr_coef_int [cmp][pivot_idx][i][j]	se(v)
mmr_coef [cmp][pivot_idx][i][j]	u(v)
} // the j-th coefficients	
} // the i-th order	
} // MMR coefficients	
} // pivot_idx	
} // cmp	
if (!disable_residual_flag) {	
for (cmp = 0; cmp < 3; cmp++) { //quantization parameters	
nlq_offset [cmp]	u(v)
hdr_in_max_int [cmp]	ue(v)
hdr_in_max [cmp]	u(v)
linear_deadzone_slope_int [cmp]	ue(v)
linear_deadzone_slope [cmp]	u(v)
linear_deadzone_threshold_int [cmp]	ue(v)
linear_deadzone_threshold [cmp]	u(v)
} // cmp	

Figure 71 Specification of number of bits "v" for CM_data() syntax elements with descriptor u(v)

The definitions of the header parameter values are contained in ETCCM, Section 5.3.2, "CM Header Parameter Definitions".

The definitions of the mapping parameter values are contained in ETCCM, Section 5.3.3, "CM Mapping Parameter Definitions".

Parameter `cm_alignment_zero_bit` shall be equal to 0.

18. Annex: Sample Dual-layer MPD§

Below is an example dual-layer MPD, with dual adaptation sets – both a Base layer and an Enhancement Layer. Items of note are highlighted:

```
<Period>
  <AdaptationSet subsegmentAlignment="true" subsegmentStartsWithSAP="1" frameRate="24000/1001"
  ">
    <Representation mimeType="video/mp4" codecs="hvc1.2.10000000.L150.B0" id="base-layer"
      bandwidth="14156144" width="3840" height="2160">
      <BaseURL>BL_dual_track_BC.mp4</BaseURL>
      <SegmentBase indexRange="795-1210">
        <Initialization range="0-794"/>
      </SegmentBase>
    </Representation>
    <Representation mimeType="video/mp4" codecs="dvhe.dtr" id="enhancement-layer"
      dependencyId="base-layer" bandwidth="3466528" width="1920" height="1080">
      <BaseURL>EL_dual_track_BC.mp4</BaseURL>
      <SegmentBase indexRange="704-1119">
        <Initialization range="0-703"/>
      </SegmentBase>
    </Representation>
  </AdaptationSet>
  <AdaptationSet mimeType="audio/mp4" codecs="ec-3" lang="und"
    subsegmentAlignment="true" subsegmentStartsWithSAP="1">
    <Representation id="2" bandwidth="192000">
      <AudioChannelConfiguration
        schemeIdUri="tag:dolby.com,2014:dash:audio_channel_configuration:2011" value="F801"/>
      <BaseURL>audio.mp4</BaseURL>
      <SegmentBase indexRange="652-875">
        <Initialization range="0-651"/>
      </SegmentBase>
    </Representation>
  </AdaptationSet>
</Period>
</MPD>
```

19. Externally defined terms§

adaptation set

See [\[MPEGDASH\]](#)

asset identifier

See [\[MPEGDASH\]](#)

CMAF track file

See [\[MPEGCMAF\]](#)

essential property descriptor

See [\[MPEGDASH\]](#)

index segment

See [\[MPEGDASH\]](#)

initialization segment

See [\[MPEGDASH\]](#)

supplemental property descriptor

See [\[MPEGDASH\]](#)

Conformance§

Conformance requirements are expressed with a combination of descriptive assertions and RFC 2119 terminology. The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in the normative parts of this document are to be interpreted as described in RFC 2119. However, for readability, these words do not appear in all uppercase letters in this specification.

All of the text of this specification is normative except sections explicitly marked as non-normative, examples, and notes. [\[RFC2119\]](#)

Examples in this specification are introduced with the words “for example” or are set apart from the normative text

with `class="example"`, like this:

EXAMPLE 20

This is an example of an informative example.

Informative notes begin with the word "Note" and are set apart from the normative text with `class="note"`, like this:

Note, this is an informative note.

Index

Terms defined by this specification

[adaptation set](#)

[addressing modes](#)

[alternative content](#)

[asset identifier](#)

[associated content](#)

[audio adaptation set](#)

[availability window](#)

[available](#)

[bitstream switching adaptation set](#)

[CMAF track file](#)

[dynamic MPD](#)

[effective time shift buffer](#)

[essential property descriptor](#)

[explicit addressing](#)

[indexed addressing](#)

[index segment](#)

[initialization segment](#)

[IOP](#)

[live edge](#)

[main content](#)

[Media Presentation](#)

[media segment](#)

[MPD](#)

[MPD refreshes](#)

[MPD start time](#)

[MPD timeline](#)

[MPD validity duration](#)

[period-connected](#)

[periods](#)

[presentation delay](#)

[representation](#)

[sample timeline](#)

[segment](#)

[segment availability times](#)

[segment end point](#)

[segment references](#)

[segment start point](#)

[simple addressing](#)

[static MPD](#)
[supplemental property descriptor](#)
[text adaptation set](#)
[thumbnail adaptation set](#)
[timescale](#)
[timescale units](#)
[time shift](#)
[time shift buffer](#)
[unnecessary segment reference](#)
[video adaptation set](#)
[XLink](#)
[XLink elements](#)

References§

Normative References§

[CEA708]

[Digital Television \(DTV\) Closed Captioning CEA-708-E](#). URL:
https://standards.cta.tech/kwspub/published_docs/ANSI-CTA-708-E-Preview.pdf

[Dolby-TrueHD]

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

[DolbyVision-ISOBMFF]

[Dolby Vision streams within the ISO base media file format](#). URL:
<https://www.dolby.com/us/en/technologies/dolby-vision/dolby-vision-bitstreams-within-the-iso-base-media-file-format.pdf>

[DTS9302J81100]

Implementation of DTS Audio in Media Files Based on ISO/IEC 14496.

[DTS9302K62400]

Implementation of DTS Audio in Dynamic Adaptive Streaming over HTTP (DASH).

[DVB-DASH]

[ETSI TS 103 285 V1.2.1 \(2018-03\): Digital Video Broadcasting \(DVB\); MPEG-DASH Profile for Transport of ISO BMFF Based DVB Services over IP Based Networks](#). March 2018. Published. URL:
http://www.etsi.org/deliver/etsi_ts/103200_103299/103285/01.02.01_60/ts_103285v010201p.pdf

[EME-INITDATA-CENC]

David Dorwin; et al. "[cenc](#)" Initialization Data Format. 15 September 2016. NOTE. URL:
<https://www.w3.org/TR/eme-initdata-cenc/>

[ENCRYPTED-MEDIA]

David Dorwin; et al. [Encrypted Media Extensions](#). 18 September 2017. REC. URL:
<https://www.w3.org/TR/encrypted-media/>

[ETSI102114]

[ETSI TS 102 114: DTS Coherent Acoustics; Core and Extensions with Additional Profiles](#). URL:
https://www.etsi.org/deliver/etsi_ts/102100_102199/102114/01.04.01_60/ts_102114v010401p.pdf

[ETSI102366]

[ETSI TS 102 366: Digital Audio Compression \(AC-3, Enhanced AC-3\) Standard](#). URL:
https://www.etsi.org/deliver/etsi_ts/102300_102399/102366/01.04.01_60/ts_102366v010401p.pdf

[ETSI103190-1]

[ETSI TS 103 190-1 V1.3.1 \(2018-02\): Digital Audio Compression \(AC-4\) Standard; Part 1: Channel based coding](#). February 2018. Published. URL:
http://www.etsi.org/deliver/etsi_ts/103100_103199/10319001/01.03.01_60/ts_10319001v010301p.pdf

[ETSI103433-1]

[ETSI TS 103 433-1: High-Performance Single Layer High Dynamic Range \(HDR\) System for use in Consumer Electronics devices; Part 1: Directly Standard Dynamic Range \(SDR\) Compatible HDR System \(SL-HDR1\)](#). URL:
https://www.etsi.org/deliver/etsi_ts/103400_103499/10343301/01.02.01_60/ts_10343301v010201p.pdf

[ISO14496-15]

[Information technology -- Coding of audio-visual objects -- Part 15: Carriage of network abstraction layer \(NAL\) unit structured video in the ISO base media file format](#). February 2017. Published. URL:
<https://www.iso.org/standard/69660.html>

[ISO14496-3-2009-AMD4-2013]

[New levels for AAC profiles](https://www.iso.org/standard/63022.html). December 2013. Published. URL: <https://www.iso.org/standard/63022.html>

[ISO14496-30]

[Information technology -- Coding of audio-visual objects -- Part 30: Timed text and other visual overlays in ISO base media file format](https://www.iso.org/standard/75394.html). November 2018. Published. URL: <https://www.iso.org/standard/75394.html>

[ISO23000-19-2018-AMD2-2019]

[XHE-AAC and other media profiles](https://www.iso.org/standard/74442.html). January 2019. Published. URL: <https://www.iso.org/standard/74442.html>

[ISO23001-8]

[Information technology -- MPEG systems technologies -- Part 8: Coding-independent code points](https://www.iso.org/standard/69661.html). May 2016. Withdrawn. URL: <https://www.iso.org/standard/69661.html>

[ISO23003-1]

[Information technology -- MPEG audio technologies -- Part 1: MPEG Surround](https://www.iso.org/standard/44159.html). February 2007. Published. URL: <https://www.iso.org/standard/44159.html>

[ISO23008-3]

[Information technology -- High efficiency coding and media delivery in heterogeneous environments -- Part 3: 3D audio](https://www.iso.org/standard/74430.html). February 2019. Published. URL: <https://www.iso.org/standard/74430.html>

[ISOBMFF]

[Information technology -- Coding of audio-visual objects -- Part 12: ISO Base Media File Format](http://standards.iso.org/ittf/PubliclyAvailableStandards/c068960_ISO_IEC_14496-12_2015.zip). December 2015. International Standard. URL: http://standards.iso.org/ittf/PubliclyAvailableStandards/c068960_ISO_IEC_14496-12_2015.zip

[ITU-R-BT.709]

[Recommendation ITU-R BT.709-6 \(06/2015\), Parameter values for the HDTV standards for production and international programme exchange](https://www.itu.int/rec/R-REC-BT.709/). 17 June 2015. Recommendation. URL: <https://www.itu.int/rec/R-REC-BT.709/>

[MEDIA-FRAGS]

Raphaël Troncy; et al. [Media Fragments URI 1.0 \(basic\)](https://www.w3.org/TR/media-frags/). 25 September 2012. REC. URL: <https://www.w3.org/TR/media-frags/>

[MIXED-CONTENT]

Mike West. [Mixed Content](https://www.w3.org/TR/mixed-content/). 2 August 2016. CR. URL: <https://www.w3.org/TR/mixed-content/>

[MP4]

[Information technology -- Coding of audio-visual objects -- Part 14: MP4 file format](https://www.iso.org/standard/75929.html). November 2018. Published. URL: <https://www.iso.org/standard/75929.html>

[MPEG2TS]

[Information technology -- Generic coding of moving pictures and associated audio information -- Part 1: Systems](https://www.iso.org/standard/75928.html). June 2019. Published. URL: <https://www.iso.org/standard/75928.html>

[MPEG4]

ISO/IEC 14496-12: ISO base media file format. ISO/IEC.

[MPEGAAC]

[Information technology -- Coding of audio-visual objects -- Part 3: Audio](https://www.iso.org/standard/53943.html). September 2009. Published. URL: <https://www.iso.org/standard/53943.html>

[MPEGAVC]

[Information technology -- Coding of audio-visual objects -- Part 10: Advanced Video Coding](https://www.iso.org/standard/66069.html). September 2014. Published. URL: <https://www.iso.org/standard/66069.html>

[MPEGCENC]

[Information technology -- MPEG systems technologies -- Part 7: Common encryption in ISO base media file format files](https://www.iso.org/standard/68042.html). February 2016. Published. URL: <https://www.iso.org/standard/68042.html>

[MPEGCMAF]

[Information technology -- Multimedia application format \(MPEG-A\) -- Part 19: Common media application format \(CMAF\) for segmented media](https://www.iso.org/standard/71975.html). January 2018. Published. URL: <https://www.iso.org/standard/71975.html>

[MPEGDASH]

[Information technology -- Dynamic adaptive streaming over HTTP \(DASH\) -- Part 1: Media presentation description and segment formats](https://www.iso.org/standard/75485.html). August 2019. Published. URL: <https://www.iso.org/standard/75485.html>

[MPEGDASHCMAFPROFILE]

N18641 WD of ISO/IEC 23009-1 4th edition AMD 1 Client event and timed metadata processing.

[MPEGHEVC]

[Information technology -- High efficiency coding and media delivery in heterogeneous environments -- Part 2: High efficiency video coding](https://www.iso.org/standard/69668.html). October 2017. Published. URL: <https://www.iso.org/standard/69668.html>

[RFC2119]

S. Bradner. [Key words for use in RFCs to Indicate Requirement Levels](https://tools.ietf.org/html/rfc2119). March 1997. Best Current Practice. URL: <https://tools.ietf.org/html/rfc2119>

[RFC7232]

R. Fielding, Ed.; J. Reschke, Ed.. [Hypertext Transfer Protocol \(HTTP/1.1\): Conditional Requests](https://httpwg.org/specs/rfc7232.html). June 2014. Proposed Standard. URL: <https://httpwg.org/specs/rfc7232.html>

[RFC7233]

R. Fielding, Ed.; Y. Lafon, Ed.; J. Reschke, Ed.. [Hypertext Transfer Protocol \(HTTP/1.1\): Range Requests](#). June 2014. Proposed Standard. URL: <https://httpwg.org/specs/rfc7233.html>

[RFC8446]

E. Rescorla. [The Transport Layer Security \(TLS\) Protocol Version 1.3](#). August 2018. Proposed Standard. URL: <https://tools.ietf.org/html/rfc8446>

[SCTE128-1]

ANSI/SCTE 128-1: AVC Video Constraints for Cable Television Part 1 - Coding. URL: https://www.scte.org/SCTEDocs/Standards/ANSI_SCTE%20128-1%202018.pdf

[XLINK]

Steven DeRose; Eve Maler; David Orchard. [XML Linking Language \(XLink\) Version 1.0](#). 27 June 2001. REC. URL: <https://www.w3.org/TR/xlink/>

Informative References

[3GPP26.116]

3GPP TS 26.116 (03/2016): Television (TV) over 3GPP services; Video Profiles..

[ATSC3]

ATSC Standard: A/300:2017 "ATSC3.0 System". URL: <https://www.atsc.org/wp-content/uploads/2017/10/A300-2017-ATSC-3-System-Standard-1.pdf>

[EBU-TT]

EBU TECH 3350: "EBU-TT Subtitling format definition". URL: <https://tech.ebu.ch/docs/tech/tech3350.pdf>

[ECMASCRIPT]

ECMAScript Language Specification. URL: <https://tc39.github.io/ecma262/>

[MEDIA-SOURCE]

Matthew Wolenetz; et al. [Media Source Extensions™](#). 17 November 2016. REC. URL: <https://www.w3.org/TR/media-source/>

[SCTE214-1]

ANSI/SCTE 214-1 2016: MPEG DASH for IP-Based Cable Services Part 1: MPD Constraints and Extensions. 2016. URL: http://scte.org/SCTEDocs/Standards/ANSI_SCTE%20214-1%202016.pdf

[SMPTE2052-1-2013]

SMPTE ST 2052-1:2013 "Timed Text Format (SMPTE-TT)". URL: <https://doi.org/10.5594/SMPTE.ST2052-1.2013>

[SMPTE2052-10]

SMPTE 2052-10: Conversion from CEA-608 Data to SMPTE-TT. URL: <https://www.smpte.org/sites/default/files/RP2052-10-2013.pdf>

[SMPTE2052-11]

Conversion from CEA-708 Caption Data to SMPTE-TT. URL: <https://www.smpte.org/sites/default/files/RP2052-11-2013.pdf>

[TTML-IMSC1.1]

Pierre-Anthony Lemieux. [TTML Profiles for Internet Media Subtitles and Captions 1.1](#). 8 November 2018. REC. URL: <https://www.w3.org/TR/ttml-imsc1.1/>

[TTML2]

Glenn Adams; Cyril Concolato. [Timed Text Markup Language 2 \(TTML2\)](#). 8 November 2018. REC. URL: <https://www.w3.org/TR/ttml2/>

Issues Index

ISSUE 1 Need to add a paragraph on interoperability on baseline, if we have any ↵

ISSUE 2 We could benefit from some detailed examples here, especially as clock sync is such a critical element of live services. ↵

ISSUE 3 What about [period](#) connectivity? [#238](#) ↵

ISSUE 4 Update to match [\[MPEGDASH\]](#) 4th edition. ↵

ISSUE 5 We need to clarify how to determine the right value for `SAP_type`. [#235](#) ↵

ISSUE 6 Once we have a specific `@earliestPresentationTime` proposal submitted to MPEG we need to update this section to match. See [#245](#). This is now done in [\[MPEGDASH\]](#) 4th edition - need to synchronize this text. [↵](#)

ISSUE 7 What exactly is metadata `@codecs` supposed to be? <https://github.com/Dash-Industry-Forum/DASH-IF-IOP/issues/290> [↵](#)

ISSUE 8 An illustration here would be very useful. [↵](#)

ISSUE 9 <https://github.com/Dash-Industry-Forum/DASH-IF-IOP/issues/284> [↵](#)

ISSUE 10 Why is the above a SHOULD? If it matters enough to signal, we should make it SHALL? <https://github.com/Dash-Industry-Forum/DASH-IF-IOP/issues/286> [↵](#)

ISSUE 11 This chapter already includes changes from [#274](#) [↵](#)

ISSUE 12 <https://github.com/Dash-Industry-Forum/DASH-IF-IOP/issues/333> [↵](#)

ISSUE 13 What do relative BaseURLs do? Do they just incrementally build up the URL? Or are they ignored? This algorithm leaves it unclear, only referencing absolute BaseURLs. We should make it explicit. [↵](#)

ISSUE 14 The text here is technically correct but could benefit from being reworded in a simpler and more understandable way. If anyone finds themselves with the time, an extra pass over this would be helpful. [↵](#)

ISSUE 15 We need to clarify how to determine the right value for `startsWithSAP`. [#235](#) [↵](#)

ISSUE 16 Add a reference here to help readers understand what are "IDS-like SAPs (i.e. SAPs of type 1 or 2)". [↵](#)

ISSUE 17 Allowing `Representation@codecs` to be absent might make it more difficult to make bitstream-switching-oblivious clients. If we require `Representation@codecs` to always be present, client developer life could be made simpler. [↵](#)

ISSUE 18 What is the above talking about? [↵](#)

ISSUE 19 This section could use another pass to make it easier to read. [↵](#)

ISSUE 20 How do leap seconds tie into this? See [#161](#) [↵](#)

ISSUE 21 <https://github.com/Dash-Industry-Forum/DASH-IF-IOP/issues/274> [↵](#)

ISSUE 22 insert reference to encryption. [↵](#)

ISSUE 23 This and everything below needs to be updated to conform to timing model. [↵](#)

ISSUE 24 Needs proper Bikeshed formatting and references [↵](#)

ISSUE 25 Check and align references in original text. [↵](#)

ISSUE 26 Is there something that goes into more depth about 404s? These statements need a better home. [↵](#)

ISSUE 27 Needs to be checked for conformance with timing model. [↵](#)

ISSUE 28 Needs proper Bikeshed formatting and referencing [↵](#)

ISSUE 29 Needs deduplication of DASH concepts that are re-defined here. [↵](#)

ISSUE 30 What are "extensions"? Move this to features/constraints chapters? [↵](#)

ISSUE 31 Where does UHD fit? Why is it in a separate chapter? We should unify. [↵](#)

ISSUE 32 What is the correct scoping for the above requirement? Is the composition time requirement specific to H.264/H.265? Or does it apply to all bitstream switching video? Or does it apply to all bitstream switching, not only video? [↵](#)

ISSUE 33 What does the above requirement actually mean - what does an implementation have to do? Unclear right now. [↵](#)

ISSUE 34 ITU-T T.35 referenced above seems unrelated to the topic. What is the correct reference? [↵](#)

ISSUE 35 Is the `payload_type` sentence meant to be a requirement or a description of the referenced spec or what is the utility of this statement in IOP? [↵](#)

ISSUE 36 There is a bunch of stuff below with no obvious connection to UHD. Should this not also be in the non-UHD HEVC chapter? [↵](#)

ISSUE 37 Clients in browsers must assume what the CDM support as there is no standardized API for probing the platform for knowing which Common Encryption protection scheme is supported. A bug is open on W3C EME and a pull request exists [here](#) for the ISOBMFF file format bytestream and a proposal is open for probing the platform on the encryption mode supported. [↵](#)

ISSUE 38 In this context, it is possible that the several quality levels are available under the same license right. Add text explaining why a shall is the way to do. [↵](#)

ISSUE 39 This seems like an unlikely problem in real client implementations. Do we know of clients that actually exhibit the problematic behavior? Look at EME and define if this is still a problem. Take advantage of the meeting in May with W3C [↵](#)

ISSUE 40 To be reviewed in light of CMAF and segment/chunk and low latency. [↵](#)

ISSUE 41 For key hierarchy, add a sentence explaining that mixing DRM systems is possible with system constraints. [↵](#)

ISSUE 42 To be completed. Look at encryption: Key available for license server "early" for been able to generate licenses (root or leaf licenses). Avoid the license server been on the critical path. Encourage license persistence in the client. [↵](#)

ISSUE 43 Merge Annex B from 4.3 to live services chapter (if not already duplicated). [↵](#)

ISSUE 44 Where is this used? Why is it an annex? Consider restructuring to improve usefulness. [↵](#)

ISSUE 45 Where is this used? Why is it an annex? Consider restructuring to improve usefulness. [↵](#)

ISSUE 46 Where is this used? Why is it an annex? Consider restructuring to improve usefulness. [↵](#)

ISSUE 47 Where is this used? Why is it an annex? Consider restructuring to improve usefulness. [↵](#)

ISSUE 48 Where is this used? Why is it an annex? Consider restructuring to improve usefulness. [↵](#)

