DASH-IF implementation guidelines: the DASH timing model
Commit Snapshot, 1 October 2019

This version:

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 Timing model
5.1 Conformance requirements
5.2 MPD Timeline
5.3 Periods
5.4 Representations
5.5 Sample timeline
5.6 Clock drift is forbidden
5.7 Media segments
5.7.1 Media segment duration deviation
5.7.2 Segments must be aligned
5.8 Period connectivity
5.8.1 Period continuity
5.9 Dynamic MPDs
5.9.1 Real time clock synchronization
5.9.2 Availability
5.9.3 Time shift buffer
5.9.4 Presentation delay
5.9.5 MPD updates
5.9.5.1 Adding content to the MPD
5.9.5.2 Removing content from the MPD
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.
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 [MPEGMAF], [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' [MPEGCMAP] 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 [MPEGCMAP] 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.

_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.

![Diagram](image)

*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 where 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.

<table>
<thead>
<tr>
<th>MPEGDASH</th>
<th>MPEGCMAF</th>
<th>ISOBMFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>(media) segment, subsegment</td>
<td>CMAF segment</td>
<td></td>
</tr>
<tr>
<td>initialization segment</td>
<td>CMAF header</td>
<td></td>
</tr>
<tr>
<td>index segment, segment index</td>
<td></td>
<td>segment index box (sidx)</td>
</tr>
</tbody>
</table>

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. Timing model

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.

5.1. 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. 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 cope 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 §5.9.5 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.

![Image](image-url) **Figure 4** The primary elements described by an MPD.

### 5.3. 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.

![Image](image-url) **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.
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 § 5.9.5 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.

5.4. 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](image)

**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.
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.
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.

### 5.5. 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).

![Sample timelines](image)

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 \texttt{@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.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure10.png}
\caption{\texttt{@presentationTimeOffset} is the key component in establishing the relationship between the MPD timeline and a sample timeline.}
\end{figure}

The point on the sample timeline indicated by \texttt{@presentationTimeOffset} is equivalent to the period start point on the MPD timeline ([MPEGDASH] Table 15). The value is provided by SegmentTemplate\texttt{@presentationTimeOffset} or SegmentBase\texttt{@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 \texttt{MpdTime} = Period\texttt{@start} + (SampleTime - \texttt{@presentationTimeOffset}) / \texttt{@timescale}.

5.6. 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.

5.7. 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.

[MPEGCMAF] 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 § 5.13.4.1 *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**.

### 5.7.1. 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.

![Diagram](image)

*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]**.

### 5.7.2. 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
5.8. 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).

![Diagram showing period connectivity](image)

**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.

![Diagram showing two periods with overlapping segments](image)

**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.

### 5.8.1. Period continuity

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.
5.9. Dynamic MPDs

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.

5.9.1. Real time clock synchronization

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]:

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.
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. The segment end point is 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 criterion 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.

<table>
<thead>
<tr>
<th>Not available</th>
<th>Available</th>
<th>Available</th>
<th>Available</th>
<th>Available</th>
<th>Available</th>
<th>Not available</th>
</tr>
</thead>
</table>

**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.

5.9.3. 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. § 5.9.2 Availability - not every media segment in the time shift buffer is guaranteed to be available.
2. § 5.9.4 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.

<table>
<thead>
<tr>
<th>Not presentable</th>
<th>Potentially presentable</th>
<th>Potentially presentable</th>
<th>Potentially presentable</th>
<th>Potentially presentable</th>
<th>Not presentable</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPD@timeShiftBufferDepth</td>
<td>Time shift buffer</td>
<td>Now</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

5.9.4. 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.

![Diagram](image)

*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.

A common error in DASH content authoring is to attempt to use MPD@minBufferTime to control the presentation delay. This attribute describes the jitter in content encoding and is determined by the encoder or derived from the encoder configuration.

5.9.5. 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.
**5.9.5.1. 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.

![Diagram](image)

*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.

**5.9.5.2. 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 § 5.9.2 Availability.
To determine the content that may be removed, let $\text{EarliestRemovalPoint}$ be availability window end + $\text{MPD}\text{@minimumUpdatePeriod}$.

Note: As each representation has its own availability window, so does each representation have its own EarliestRemovalPoint.

![Diagram](image)

**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 $@r \neq 0$) 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.

### 5.9.5.3. 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.

### 5.9.6. 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.

### 5.9.6.1. 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.

5.10. Timing of stand-alone IMSC1 and WebVTT text files

Some services store text adaptation sets in stand-alone IMSC1 or WebVTT files, without segmentation or [ISOBMF] 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.

```xml
<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.

5.11. Forbidden techniques

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.

5.12. Examples

This section is informative.

5.12.1. Offer content with imperfectly aligned tracks

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.](image)
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 representation 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.

![Diagram](image)

**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.

### 5.12.2. Split a period

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.
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 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 § 5.13.4.2 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.

5.12.3. 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.

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

### 5.13. 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.

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.

### 5.13.1. 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].

---

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.

---

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

```xml
<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.

5.13.2. Structure of the index segment

The **index segment** SHALL consist of a single Segment Index Box (**sidx**) as defined by [ISO/IEC 14496-12]. The field layout is as follows:

```plaintext
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 4** We need to clarify how to determine the right value for SAP_type. #235

### 5.13.2.1. Moving the period start point (indexed 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 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.

### 5.13.3. 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.
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 unique 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.

```xml
<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.
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 § 5.4 Representations and § 5.9.5.2 Removing content from the MPD to understand the constraints that apply to segment reference removal.
5.13.4. Simple addressing

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 § 5.13.4.1 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).
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.

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 §5.7 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.

5.13.4.2. 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 §5.13.4.3 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:
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 SegmentCount = Ceiling(AsSeconds(Period@duration) / (SegmentTemplate@duration / 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.

```xml
<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" />
      </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.

```xml
<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.

5.14. 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 \(2^{53}\).

5.15. 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
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. Externally defined terms

adaptation set
See [MPEGDASH]

CMAF track file
See [MPEGCMAF]

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 9
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
availability window
available
CMAF track file
References

Normative References

[DVB-DASH]

[ISOBMFF]

[MPEG2TS]
We could benefit from some detailed examples here, especially as clock sync is such a critical element of live services.

What about period connectivity? #238


We need to clarify how to determine the right value for SAP_type. #235

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.