DASH-IF implementation guidelines: restricted timing model

Commit Snapshot, 24 October 2024

This version:
https://dashif.org/Guidelines-TimingModel/

Issue Tracking:
GitHub
GitHub
Inline In Spec

Editor:
DASH Industry Forum

Table of Contents

§ 1. Purpose

The guidelines defined in this document support the creation of interoperable services for high-quality video distribution based on MPEG-DASH and related standards. These guidelines are provided in order to address DASH-IF members' needs and industry best practices. The guidelines support the implementation of conforming service offerings as well as DASH client implementations.

The restricted timing model constrains the ordinary timing model defined in [DASH] primarily by disallowing gaps in presentations, thereby increasing the compatibility of DASH services with devices that do not have robust support for playback of content with gaps.

This timing model also provides editorial flexibility for the presentation author by allowing new periods to be started at any point. This is achieved by:

1. Strictly defining the period boundary rules on the DASH service side.
2. Permitting fleixble behavior from clients in how they transition between periods (to account for implementation limitations).

NOTE: Some alternative timing model interpretations significantly restrict the ability of content authors to define period boundaries.

In addition to defining the constraints for a restricted timing model, this document attempts to explain and illustrate many DASH timing concepts that often cause confusion, without constraining them further than already done by DASH-IF general guidelines, [DASH] or [CMAF].

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.

This part of the DASH-IF implementation guidelines is published as a stand-alone document for editorial reasons. Refer to the master document to understand the context in which this document should be viewed.

§ 2. Interpretation

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

If a **service provider** follows these requirements in a published DASH service, the published DASH 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.

If a **client implementer** follows the client-oriented requirements described in this document, the DASH client will play content conforming to this document provided that the client device media platform supports all features used by a particular DASH service (e.g. the codecs and DRM systems).

This document uses statements of fact when describing normative requirements defined in referenced specifications such as [DASH] and [CMAF]. References are typically provided to indicate where the requirements are defined.

[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.

EXAMPLE 1 ¶
Statement of fact:

- A DASH presentation **is** a sequence of consecutive non-overlapping periods [DASH].

New or more constrained requirement:

- Segments **SHALL NOT** use the MPEG-TS container format.

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

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 (dynamic adaptive streaming over HTTP) [DASH] is a technology for adaptive media delivery. Initially published by ISO/IEC in April 2012, it has been continually updated, with the 4th edition published in 2020.

CMAF (common media application format) [CMAF] is a media container format based on ISO Base Media File Format [ISOBMFF]. It defines data structures for media delivery compatible with DASH and other similar technologies such as [HLS]. Initially published by ISO/IEC in 2018, it has been updated in 2019 with the publishing of the 2nd edition.

This document is based on the 4th edition DASH [DASH] and 2nd edition CMAF [CMAF] 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.

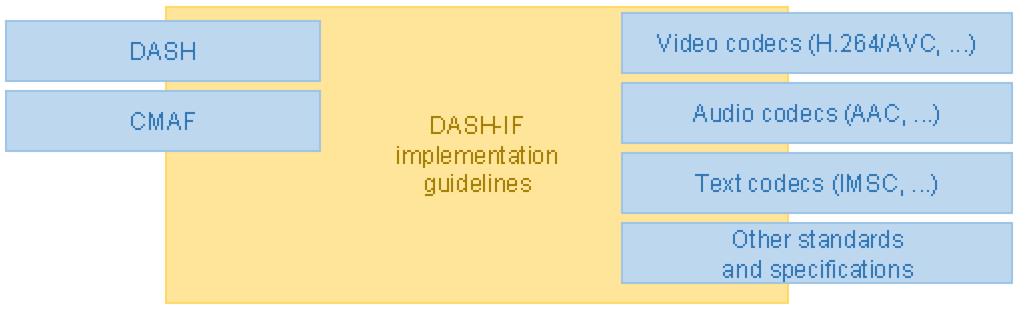
DASH
CMAF
DASH-IF implementation guidelines
Video codecs (H.264/AVC, ...)
Audio codecs (AAC, ...)
Text codecs (IMSC, ...)
Other standards and specifications

Figure 1 *This document connects DASH with international standards and industry specifications.*

[DASH] defines a highly flexible set of building blocks that needs to be constrained to ensure interoperable behavior in common scenarios. The necessary media container constraints are largely defined by [CMAF] and [DASH-CMAF]. This document defines further constraints to limit DASH features to those that are considered appropriate for use in interoperable clients and services.

Clients consuming DASH content will need to interact with the host device’s media platform. The guidelines in this document assume that the media platform implements APIs that are equivalent to Media Source Extensions [media-source] and Encrypted Media Extensions [encrypted-media]. API level compatibility is not required but equivalent features are expected.

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].

§ 4.1. Structure of a DASH presentation

[DASH] 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 during playback of a presentation in order to obtain media samples.

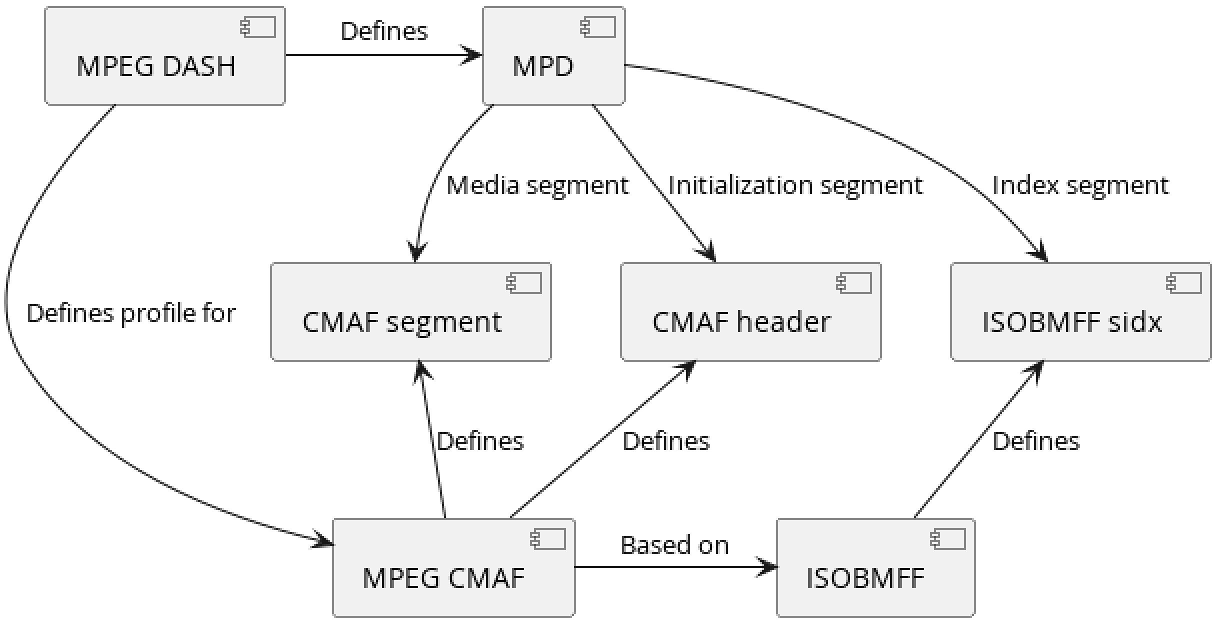

Figure 2 *Relationships of primary DASH data structures and the standards they are defined in.*

The MPD is an XML file that follows a schema defined in [DASH]. Various 3rd party extension points are defined in the XML schema. This document defines some extensions, as do other industry specifications.

[DASH] 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.

[CMAF] is a constrained media format based on [ISOBMFF], specifically designed for adaptive streaming. This document requires the use of [CMAF] compatible data containers. The requirements for the usage of CMAF with DASH are defined by [DASH-CMAF].

NOTE: The relationship to [CMAF] is constrained to the container format, as primarily expressed by [DASH-CMAF]. In particular, there is no requirement to conform to [CMAF] media profiles.

The data container format defines the physical structure of the following components of a presentation:

1. Each representation contains an initialization segment.
2. Each representation contains any number of media segments.
3. Some representations may contain an index segment, depending on the addressing mode used.

NOTE: HLS (HTTP Live Streaming) [HLS] is an adaptive media delivery technology similar to DASH that also supports CMAF. Under certain constraints, content conforming to CMAF can be delivered to clients using both DASH and HLS.

§ 4.2. Terminology cross-reference across standards

Different documents often use different terms to refer to the same structural components of DASH presentations. A quick cross-reference of terms commonly found causing confusion is presented here:

[DASH]	**[CMAF]**	**[ISOBMFF]**
(media) segment, subsegment	CMAF segment, CMAF fragment	
initialization segment	CMAF header	
index segment, segment index		segment index box (`sidx`)

Figure 3 *Cross-reference of closely related terms in different standards.*

§ 4.3. Terminology choices in this document

This document is intended to be a set of guidelines easily understood by solution designers and developers. In the interest of ease of understanding, some important adjustments in terminology are made compared to the underlying standards, described here.

[DASH] has the concept of "segment" (URL-addressable media object) and "subsegment" (byte range of URL-addressable media object), whereas [CMAF] does not make such a distinction. This document uses [CMAF] terminology, with the term "segment" in this document being equivalent to "CMAF segment". The term "segment" in this document may be equivalent to either "segment" or "subsegment" in [DASH], depending on the addressing mode used.

This document's concept of the MPD timeline is not directly expressed in [DASH]. To improve understandability of the timing model, this document splits the DASH concept of "presentation timeline" ([DASH] 7.2.1) into two separate concepts: the aggregated component (MPD timeline) and the representation specific component (sample timeline). These concepts are distinct but mutually connected via metadata in the MPD.

[DASH] uses "representation" to refer to a set of files and associated metadata, with the same "representation" possibly used in different parts of a DASH presentation and/or in different presentations. This document uses representation to refer to an individual instance of a [DASH] "representation" - a set of data in an MPD that references some files containing media samples. Using the same files in two places effecticely means using two representations, whereas in [DASH] terminology it would be valid to call that a single representation used with individual conditions/caveats that apply to each usage. The deviation in terminology is intentional as it simplifies understanding and avoids having to juggle the two concepts (as the "shared" view is typically not relevant).

§ 5. Goal of the interoperable timing model

The purpose of this document is to give a holistic overview of DASH presentation timing and segment addressing, explaining the existing building blocks and rules defined by [DASH] and adding further constraints to achieve greater interoperability between DASH services and clients.

[DASH] 4.3 and 7.2.1 define the high-level structure and timing concepts of DASH, with [DASH-CMAF] further relating them to [CMAF] concepts. The DASH-IF implementation guidelines allow considerably less flexibility in timing than provided by [DASH], constraining services to a specific set of reasonably flexible behaviors that are highly interoperable with modern client platforms.

This document defines an interoperable timing model and documents segment addressing logic suitable for interoperable use cases. Alternative interpretations of DASH timing may be equally valid from a standards conformance viewpoint.

§ 6. MPD timeline

The MPD defines the ***MPD timeline*** of a DASH presentation, which serves as the baseline for all scheduling decisions made during playback and establishes the relative timing of periods and media segments. The MPD timeline informs DASH clients on when it can download and present which media segments. The contents of an MPD are a promise by a DASH service to make specific media segments available during specific time spans described by the MPD timeline.

Values on the MPD timeline are all ultimately relative to the zero point of the MPD timeline, though possibly through several layers of indirection (e.g. period A is relative to period B, which is relative to the zero point).

The ultimate purpose of the MPD is to enable the client to obtain media samples for playback. The MPD also provides the information required for a DASH client to dynamically switch between different bitrates of the same content (in different representations) to adapt to changing network conditions.

The following MPD elements are most relevant to locating and scheduling the media samples:

1. The MPD describes consecutive periods which map data onto the MPD timeline.
2. Each period describes of one or more representations, each of which provides media samples inside a sequence of media segments located via segment references. Representations contain independent sample timelines that are mapped to the time span on the MPD timeline that belongs to the period.
3. Representations within a period are grouped into adaptation sets, which associate related representations and decorate them with metadata.

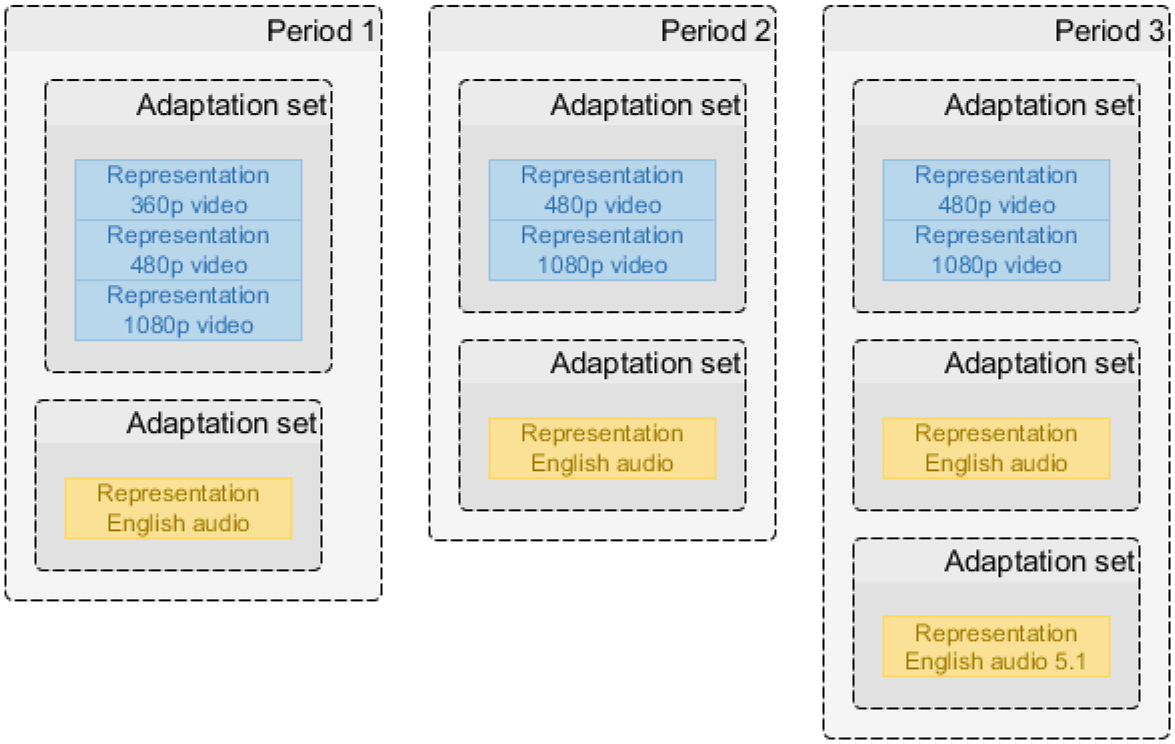
Period 1
Adaptation set
Representation 360p video
Representation 480p video
Representation 1080p video
Adaptation set
Representation English audio
Period 2
Adaptation set
Representation 480p video
Representation 1080p video
Adaptation set
Representation English audio
Period 3
Adaptation set
Representation 480p video
Representation 1080p video
Adaptation set
Representation English audio
Adaptation set
Representation English audio 5.1

Figure 4 *The primary contents of a presentation, described by an MPD.*

§ 7. Presentation timing characteristics

There exist two types of DASH presentations, indicated by `MPD@type` [DASH]:

- In a a ***static presentation*** (`MPD@type="static"`) any media segment may be presented at any time. The DASH client is in complete control over what content is presented when and the entire presentation is available at any time.
- In a ***dynamic presentation*** (`MPD@type="dynamic"`) the MPD timeline is mapped to wall clock time, with each media segment on the MPD timeline intended to be presented at a specific moment in time (with some client-chosen time shift allowed).
    - Furthermore, media segments may become available and cease to be available with the passage of time.
    - The MPD may change over time, enabling the structure of the presentation to change over time (e.g. when a new title in the presentation is offered with a different set of languages).

In a dynamic presentation, the zero point of the MPD timeline is the mapped to the point in wall clock time indicated by the ***effective availability start time***, which is formed by taking `MPD@availabilityStartTime` and applying any `LeapSecondInformation` offset

([DASH] 5.3.9.5 and 5.13). This allows a wall clock time to be associated with each media segment, indicating the moment the media segment is intended to be presented. The zero point of the MPD timeline will move when leap seconds occur ([DASH] 5.13). See also §13.2 Leap seconds.

`MPD@mediaPresentationDuration` MAY be present in an MPD. If present, it SHALL accurately match the duration between the zero point on the MPD timeline and the end of the last period, including the duration of any XLink periods. Clients SHALL calculate the total duration of a static presentation by adding up the durations of each period and SHALL NOT 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 (the contents of the XLink periods).

§ 8. Period timing

An MPD defines an ordered list of one or more consecutive non-overlapping ***periods*** ([DASH] 5.3.2). 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, though often indirectly (being relative to the previous period).

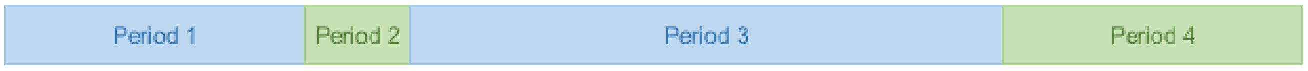

Figure 5 *An MPD defines a collection of consecutive non-overlapping periods.*

The start of a period is specified either explicitly as an offset from the MPD timeline zero point (`Period@start`) or implicitly by the end of the previous period ([DASH] 5.3.2). The duration of a period is specified either explicitly with `Period@duration` or implicitly by the start point of the next period ([DASH] 5.3.2). See also §8.1 First and last period timing in static presentations and §8.2 First and last period timing in dynamic presentations.

Periods are self-contained - a service SHALL NOT require a client 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.

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).

EXAMPLE 2 ¶
The below 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.

A period SHALL NOT have a duration of zero. MPD generators are expected to remove any periods that are, for any reason, assigned a duration of zero. This might happen, for example, due to ad insertion logic deciding not to insert any ad or due to a packager not receiving any content to insert into the period. Clients SHALL ignore periods with a duration of zero.

§ 8.1. First and last period timing in static presentations

In a static presentation, the first period SHALL start at the zero point of the MPD timeline (with a `Period@start` value of 0 seconds).

In a static presentation, the last period SHALL have a `Period@duration`.

§ 8.2. First and last period timing in dynamic presentations

In a dynamic presentation, the first period SHALL start at or after the zero point of the MPD timeline (with a `Period@start` value of 0 seconds or greater).

In a dynamic presentation, 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 presentation has an unlimited duration (that may later be shortened by an MPD update).

NOTE: A period with an unlimited duration can be converted to fixed duration by an MPD update, so even a nominally unlimited duration is effectively constrained by the MPD validity duration of the current MPD snapshot.

EXAMPLE 3 ¶
The below MPD consists of a 20-second period followed by a period of unlimited duration.

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

§ 9. Representation timing

Representations provide the content for periods. A representation is a sequence of media segments, an initialization segment, an optional index segment and related metadata ([DASH] 5.3.1 and 5.3.5).

The MPD describes each representation using a `Representation` element. For each representation, the MPD defines a set of ***segment references*** to the media segments and metadata describing the media samples provided by the representation. The segment references and much of the metadata are shared by all representations in the same adaptation set.

Each representation belongs to exactly one adaptation set and exactly one period, although a representation may be connected with a representation in another period.

EXAMPLE 4
The below MPD consists of a single 20-second period with three video, one audio and one text representation. Each representations supplies the period with 20 seconds of media samples.

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" type="static">
  <Period duration="PT20S">
    <AdaptationSet>
      <Representation id="1" mimeType="video/mp4" codecs="avc1.64001f" bandwidth="38643
      <Representation id="2" mimeType="video/mp4" codecs="avc1.640028" bandwidth="1117
      <Representation id="3" mimeType="video/mp4" codecs="avc1.640033" bandwidth="2723
    </AdaptationSet>
    <AdaptationSet lang="en">
      <Representation id="4" mimeType="audio/mp4" codecs="mp4a.40.29" bandwidth="13135
    </AdaptationSet>
    <AdaptationSet lang="en-US">
      <Representation id="5" mimeType="application/mp4" codecs="wvtt" bandwidth="428" />
    </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.

9.1. Sample timeline

The samples within a representation exist on a linear ***sample timeline*** defined by the encoder that creates the samples. Sample timelines are mapped onto the MPD timeline by metadata stored in or referenced by the MPD ([DASH] 7.3.2).

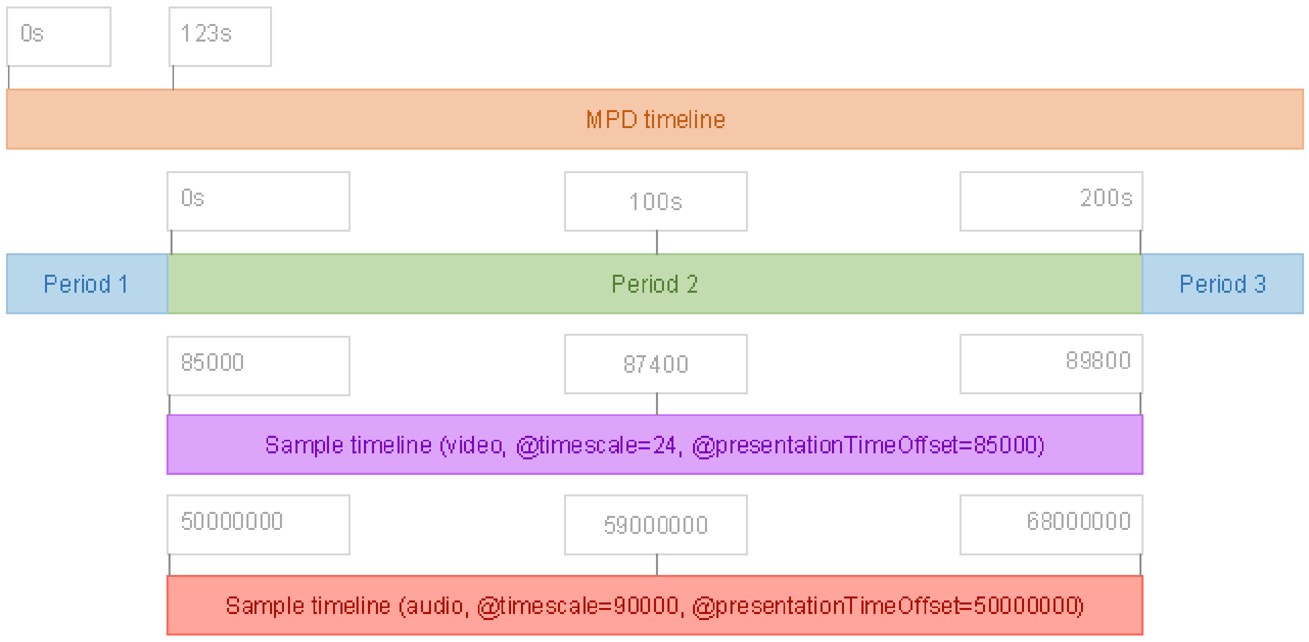
0s
123s
MPD timeline
0s
100s
200s
Period 1
Period 2
Period 3
85000
87400
89800
Sample timeline (video, @timescale=24, @presentationTimeOffset=85000)
50000000
59000000
68000000
Sample timeline (audio, @timescale=90000, @presentationTimeOffset=50000000)

Figure 6 *A sample timeline is mapped onto the MPD timeline based on parameters defined in the MPD, relating the media samples provided by a representation to the portion of the MPD timeline covered by the period that references the representation. The sample timelines extend further beyond the range of the period (full extents not illustrated).*

The sample timeline does not determine what samples are presented. It merely connects the timing of the representation to the MPD timeline and allows the correct media segments to be identified when a DASH client makes scheduling decisions driven by the MPD timeline. The exact connection between media segments and the sample timeline is defined by the addressing mode.

The same sample timeline is shared by all representations in the same adaptation set [DASH-CMAF]. Representations in different adaptation sets MAY use different sample timelines.

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 ([DASH] 7.3.2).

A sample timeline is measured in ***timescale units*** defined as a number of units per second ([DASH] 5.3.9.2 and 5.3.9.6). 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 [DASH], 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.

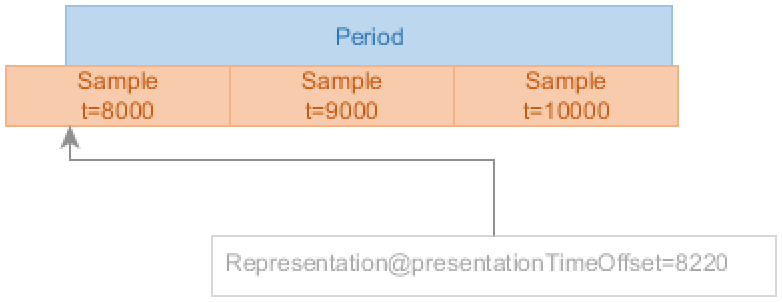

Figure 7 *`@presentationTimeOffset` is the key component in establishing the relationship between the MPD timeline and a sample timeline.*

The zero point of a sample timeline may be at the start of the period or at any earlier point. The point on the sample timeline indicated by `@presentationTimeOffset` is equivalent to the period start point on the MPD timeline ([DASH] 5.3.9.2). 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`.

§ 9.2. Referencing media segments

Each segment reference addresses a media segment that corresponds to a specific time span on the sample timeline.

The exact mechanism used to define segment references depends on the addressing mode used by the representation. All representations in the same adaptation set SHALL use the same addressing mode.

The sequence of segment references provided for a representation SHALL NOT leave gaps between media segments or define overlapping media segments.

The portion of the period that a representation must provide media segments for depends on the type of the presentation, with the requirements for each type described below.

§ 9.2.1. Necessary segment references in static presentations

In a static presentation, a representation SHALL provide enough media segments to cover the entire time span of the period.

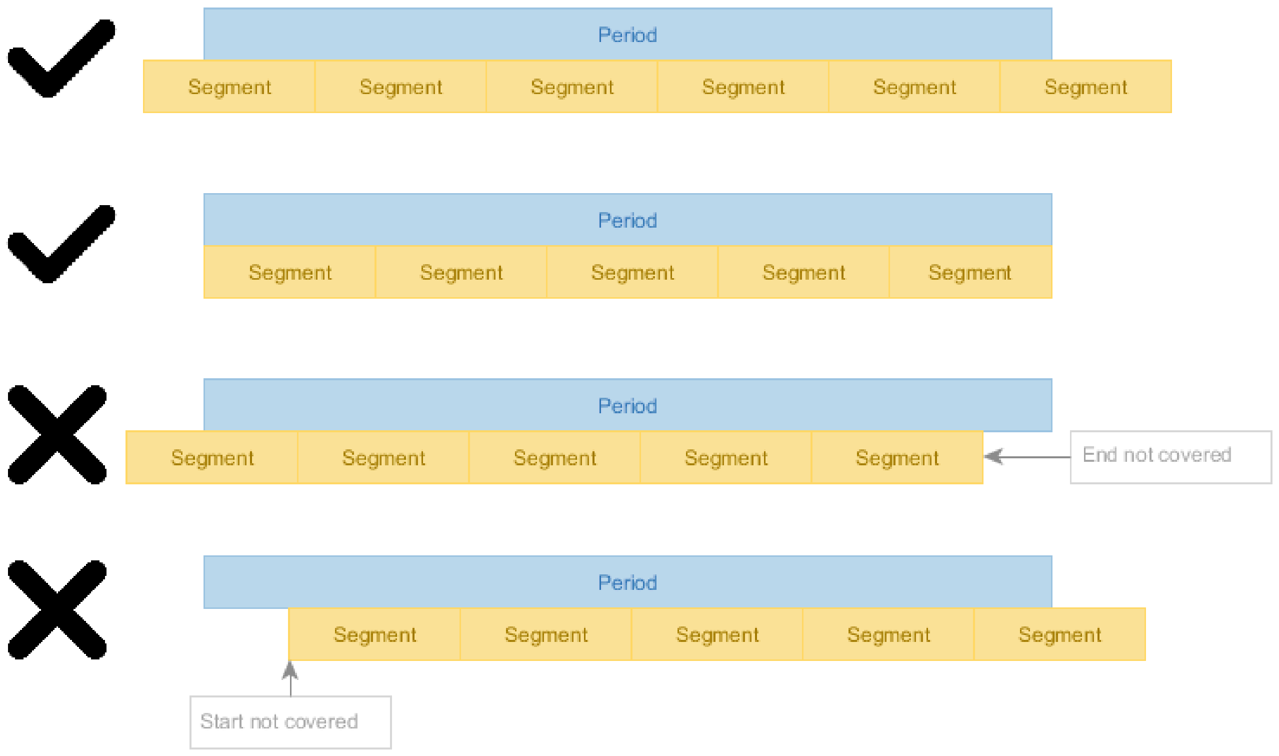

Figure 8 *In a static presentation, the entire period must be covered with media segments.*

§ 9.2.2. Necessary segment references in dynamic presentations

In a dynamic presentation, a representation SHALL provide enough media segments to cover the time span of the period that intersects with the time shift buffer at any point during the MPD validity duration.

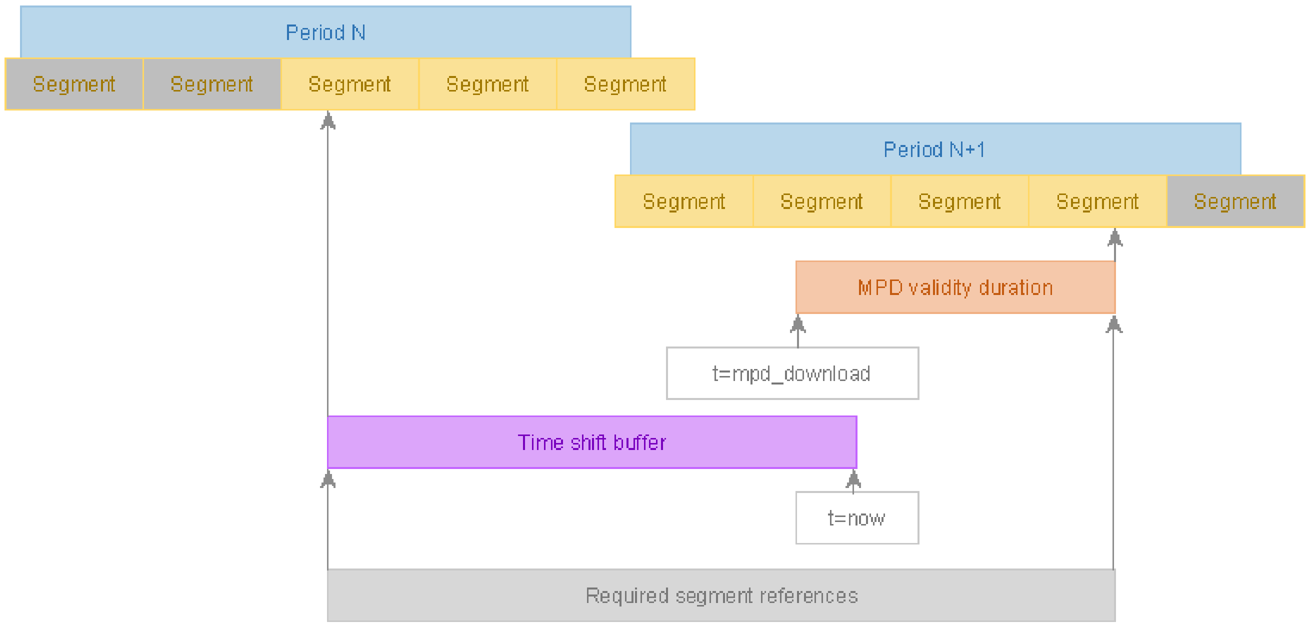

Figure 9 *In a dynamic presentation, the time shift buffer and MPD validity duration determine the set of required segment references for each representation. Media segments filled with gray need not be referenced due to falling outside the time shift buffer in its maximum extents during the MPD validity duration, despite falling within the bounds of a period.*

NOTE: In the above example, the second period is shown as extending beyond the end of the MPD validity duration (e.g. because it is of unlimited length), which effectively increases the time shift buffer to the end of the MPD validity duration. If the second period were shorter, the range of required segment references would terminate with the end of the period.

It is a valid and common situation that a media segment is required to be referenced but is not yet available. See also § 13.3 Availability.

§ 9.2.3. Removal of unnecessary segment references

An ***unnecessary segment reference*** is one that is not defined as required by § 9.2.1 Necessary segment references in static presentations or § 9.2.2 Necessary segment references in dynamic presentations.

In a static presentation, the MPD SHALL NOT contain unnecessary segment references, except for representations that use indexed addressing in which case such segment references MAY be present.

In a dynamic presentation, the MPD 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 uses `S@r` to define multiple segment references, some of which are necessary.
4. Removal of the segment reference is not allowed by content removal constraints.

§ 9.3. Alignment of periods and representations

Segment start points and segment end points do not need to be aligned with period start/end points ([DASH] 7.2.1). The general expectation is that only the content that falls within the period time span is presented by DASH clients. Allowing for overflow outside this time span ensures that periods can be easily started and ended at arbitrary positions on the MPD timeline without leaving gaps. Starting and ending periods is an editorial decision that is typically independent of the technical structure of the contents of the period.

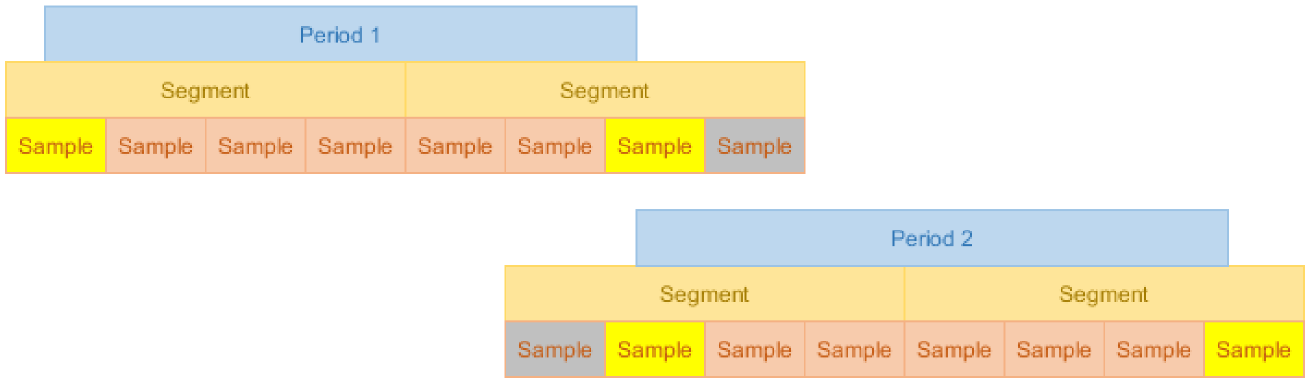

Figure 10 *Media segments and samples need not align with period boundaries. Some samples may be entirely outside a period (marked gray) and some may overlap the period boundary (yellow).*

Clients SHALL NOT present any samples from media segments that are entirely outside the period, even if such media segments are referenced.

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.

As perfect alignment between sample and period boundaries cannot be expected, clients MAY incur small time shift in either direction (within extents permitted by this document) when playing a dynamic presentation and transitioning in/out of a period where the sample and period boundaries are not aligned.

§ 10. 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.

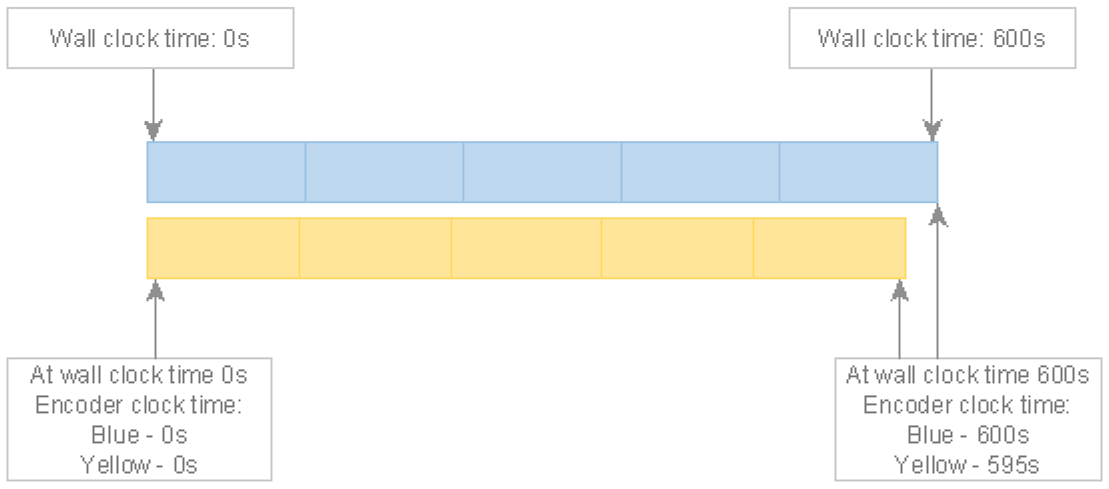

Figure 11 *Comparison of an encoder correctly tracking wall clock time (blue) and an encoder with a clock that runs 0.8% too slowly (yellow), leading it to producing fewer seconds of content than expected (the correct amount of content has been temporally compressed by the encoder to fit into a smaller number of seconds). A DASH packager cannot use the yellow encoder's output as-is or it would violate the DASH timing model, which requires services to track wall clock time, and potentially lead to track de-synchronization.*

Clock drift not only causes timing model violations when an insufficient amount of data is produced but also leads to de-synchronization of content in tracks encoded based on different clocks. [CMAF] 6.3 and 6.6.8 require tracks to be synchronized.

NOTE: A lack of data at the current wall clock time or in the past is typically a violation of the timing model, whereas there is no explicit restriction on providing data in the future.

To detect clock drift, one can check for the presence/absence of data near the current wall clock time. If data from now or the immediate past is absent, possibly the encoder has a slow clock. If data from the future is present, possibly the encoder has a fast clock. Furthermore, gradual de-synchronization of content in different tracks over a long play duration is a clear sign of clock drift on one or more of the involved encoders.

It would be unreasonable to expect DASH clients to counteract clock drift by performing their own timeline stretching or compressing during playback, even if provided with the information about clock differences. DASH clients are based on very limited media platform APIs that typically lack the capability for any such compensation. Therefore, a DASH service SHALL NOT publish content that suffers from clock drift.

The solution is to adjust the encoder so that it correctly tracks wall clock time, e.g. by performing regular small adjustments to the encoder clock to counteract any "natural" drift it may be experiencing. The exact implementation depends on the encoder timing logic and is out of scope of this document.

§ 10.1. Workarounds for clock drift

If the encoder cannot be adjusted to not suffer from clock drift, the only remaining option is to post-process its output to bring the presentation into conformance with the timing model. The facilities available to the packager are likely less powerful than those available to the encoder - it is unlikely that re-encoding/re-timing the media samples is practical in the packager. Furthermore, this type of adjustment will not eliminate track de-synchronization that will be present unless the clocks used to encode all tracks drift at the same rate.

DASH packagers are responsible for generating DASH presentations that conform to targeted standards or specifications and cannot assume perfect encoder implementations. It is a fact that some encoders suffer from clock drift. DASH packagers SHOULD implement workarounds to ensure the presentation is conforming to targeted standards and specifications. This may require some some unavoidable disruption of the end-user experience.

The following are examples of approaches a DASH packager could use to bring content from an encoder suffering clock drift into conformance:

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.

Such after-the-fact corrective actions can be disruptive and only serve as a backstop to prevent complete playback failure cased by timing model violations. Such workarounds might be satisfactory when correcting for very small drift rates, with any disruptions being relatively rare.

§ 11. Media segments

A ***media segment*** is an HTTP-addressable data structure that contains media samples, referenced by an MPD via a segment reference. The structure of a media segment is that of a CMAF segment consisting of one or more CMAF fragments [DASH-CMAF]. Different media segments may be different byte ranges accessed on the same URL.

The segment-related terminology in this document is aligned with [CMAF] rather than [DASH]. See §4.3 Terminology choices in this document to better understand the differences.

Media segments contain one or more consecutive media samples and consecutive media segments in the same representation contain consecutive media samples [CMAF].

A media segment contains the media samples that exactly match the time span on the sample timeline associated with a media segment via a segment reference ([DASH] 7.2.1 and [DASH-CMAF]), except when using simple addressing in which case a certain amount of inaccuracy may be present as defined in §18.4.1 Inaccuracy in media segment timing when using simple addressing. How segment references are defined depends on the addressing mode.

All timing-related clauses in this document refer to the nominal timing described in the MPD unless otherwise noted. DASH clients are expected to operate with nominal times in playback logic, even if the real values differ due to permitted amounts of inaccuracy.

The ***segment start point*** is the point on the MPD timeline where the media segment starts according to the segment reference obtained from the MPD. The ***segment end point*** is the segment start point plus the media segment duration defined by the segment reference.

NOTE: In [DASH] terminology, the segment start point is often equivalent to "earliest presentation time" of the media segment. However, this relation does not always hold true as "earliest presentation time" is defined in terms of media sample timing which is affected by the inaccuracy allowed under simple addressing. In contrast, the segment start point is always the nominal start point and is not affected by any potential timing inaccuracy.

Media segments in different representations of the same adaptation set are aligned ([CMAF] 7.3.4 and [DASH-CMAF]). This means they contain media samples for the same time span on the sample timeline. This is true even if using simple addressing with inaccurate media segment timing. That is, not only is the nominal timing aligned but so is the true media sample timing inside the media segments.

§ 12. Period connectivity

In certain circumstances content may be offered such that the contents of one adaptation set are technically compatible with the contents an adaptation set in a previous period ([DASH] 5.3.2.4). Such adaptation sets are ***period-connected***.

The main characteristic of connectivity is that initialization segments of representations with matching `@id` in period-connected adaptation sets are functionally equivalent ([DASH] 5.3.2.4). That is, the initialization segment of a representation in one period-connected adaptation set can be used to initialize playback of a representation with matching `@id` in the other period-connected adaptation set. Connectivity is typically achieved by using the same encoder to encode the content of multiple periods using the same settings.

In encrypted content the content key identifier `default_KID` is part of the initialization segment. Using a different content key breaks period connectivity that would otherwise exist due to matching codec configuration.

Adaptation sets SHALL NOT be signaled as period-connected if the set of representations in them is different, even if all shared representations remain compatible.

NOTE: The above constraint removes some ambiguity from the [DASH] definition, which does not explicitly state whether it is allowed to only have a subset of representations that is connected. GitHub #387

An MPD MAY contain unrelated periods between periods that contain period-connected adaptation sets.Period connectivity MAY be chained across any number of periods.

Period-connected adaptation sets content SHOULD be signaled in the MPD as period-connected. This signaling helps clients ensure seamless playback across period transitions.

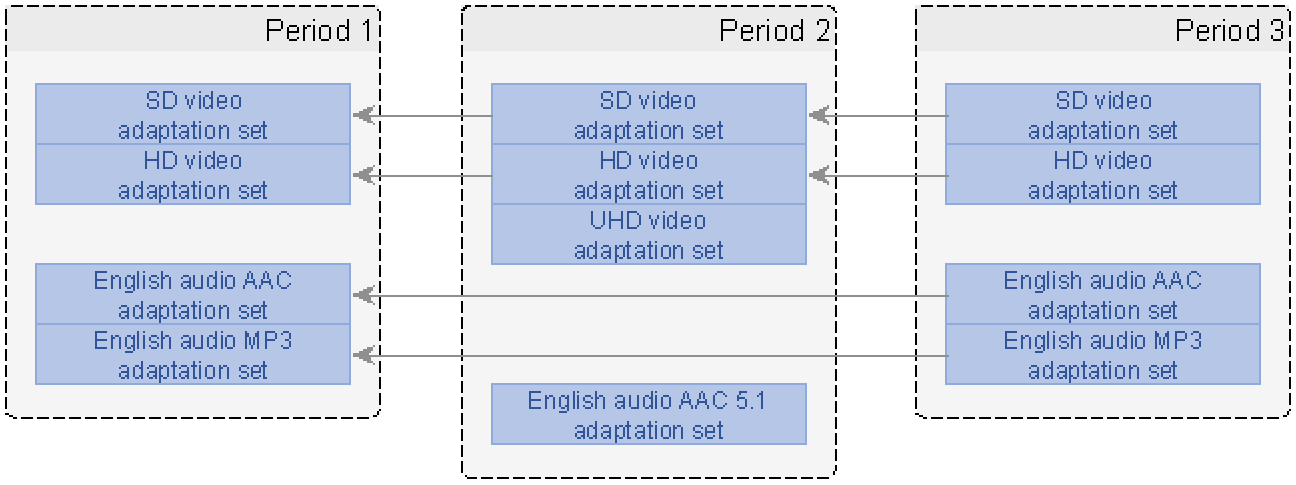

Figure 12 *Adaptation sets can be signaled as period-connected, enabling client optimizations. Arrows on diagram indicate direction of connectivity reference (from future to past), with the implied message being "the client can use the same decoder configuration it used where the arrow points to".*

The sample timelines of representations in period-connected adaptation sets MAY be discontinuous between two periods (e.g. due to encoder clock wrap-around or skipping some content as a result of editorial decisions). See also § 12.2 Period continuity.

The following signaling in the MPD indicates that two adaptation sets are period-connected across two periods ([DASH] 5.3.2.4):

- 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.

The period-connected adaptation sets have the same `@id` and the same set of `Representation@id` values ([DASH] 5.3.2.4).

§ 12.1. Segment reference duplication during connected period transitions

As a period may start and/or end in the middle of a media segment, the same media segment may simultaneously be referenced by two period-connected adaptation sets, 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.

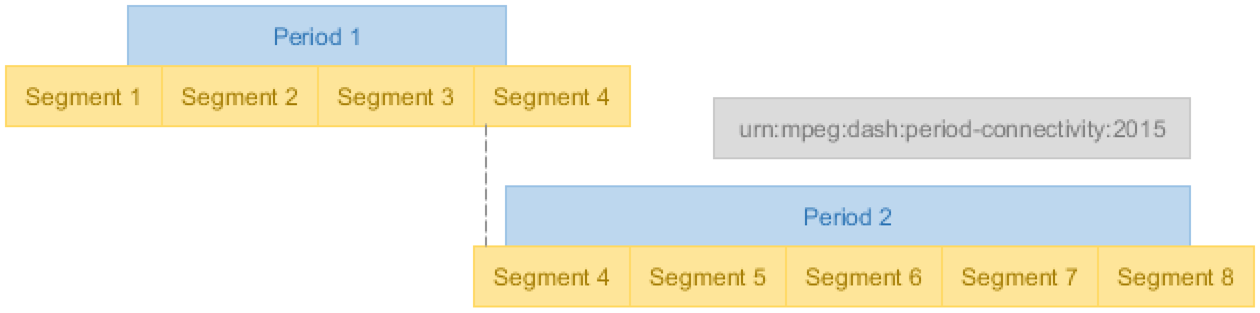

Figure 13 *The same media segment will often exist in two periods at a period-connected transition. On the diagram, this is segment 4.*

Clients SHOULD NOT present a media segment twice when it occurs on both sides of a period transition in a period-connected adaptation set.

Clients SHOULD ensure seamless playback of period-connected adaptation sets in consecutive periods. Clients unable to ensure seamless playback MAY incur some amount of time shift at the period transition point provided that the resulting time shift is permitted by the timing model.

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.

§ 12.2. Period continuity

In addition to period connectivity, [DASH] 5.3.2.4 defines period continuity. Continuity is a special case of period connectivity that indicates no timeline discontinuity is present at the transition point between the media samples of the two continuous periods. Under continuity conditions, the client is expected to be able to continue seamless playback by merely appending media segments from the new period, without any reconfiguration at the period boundary.

Continuity SHALL NOT be signaled if the first/last sample in the media segment on the period boundary does not exactly start/end on the period boundary. This cannot be expected to be generally true, as period boundaries are often an editorial decision independent of the media segment and sample layout.

NOTE: This further constrains usage of continuity compared to [DASH], which does not require the boundary samples to actually be the first/last sample in the media segment. However, that interpretation leaves room for incompatible implementations depending on how the client handles deduplication of duplicate segments at period boundaries (which would be required under the rules of the interoperable timing model in order to not leave a gap).

Period continuity MAY be signaled in the MPD when the above condition is met, in which case period connectivity SHALL NOT be simultaneously signaled on the same representation. Continuity implies connectivity ([DASH] 5.3.2.4).

The signaling of period continuity is the same as for period connectivity, except that the value to use for `@schemeIdUri` is `urn:mpeg:dash:period-continuity:2015` ([DASH] 5.3.2.4).

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.

§ 13. Dynamic presentations

The requirements in this section and its subsections only apply to dynamic presentations.

The following factors primarily differentiate dynamic presentations from static presentations:

1. The media segments of a dynamic presentation may become available over time and cease to be available after the passage of time. That is, not all segments are necessarily available at all times.
2. Playback of a dynamic presentation is synchronized to a wall clock (with some amount of client-chosen time shift allowed).
3. The MPD of a dynamic presentation may change over time, with each snapshot having a limited MPD validity duration and clients regularly downloading new snapshots of the MPD.

EXAMPLE 5 ¶

The below MPD consists of two 300-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 600 seconds.

The dynamic presentation was intended to be presented starting at 09:35 UTC on December 2, 2017, allowing for up to 400 seconds of time shift by the DASH client. By the absence of `MPD@minimumUpdatePeriod`, the MPD indicates that its contents will never change.

The absence of an `LeapSecondInformation` element indicates the service provider does not expect the service to remain accessible for long enough to encounter a leap second.

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" type="dynamic"
  availabilityStartTime="2017-12-02T09:35:00Z" timeShiftBufferDepth="PT400S">
  <Period>
    …
  </Period>
  <Period start="PT300S" duration="PT300S">
    …
  </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.

The requirements in this document mandate the removal of expired content and expired MPDs. Given the 2017 date marked in the MPD, this example is obviously expired in its entirety. The necessary removal of expired content from within the MPD has been omitted for purposes of illustration.

§ 13.1. Clock synchronization

During playback of dynamic presentations, a wall ***wall clock*** is used as the timing reference for DASH client decisions. This is a synchronized clock shared by the DASH client and service. With the exception of clock adjustments performed by the DASH client for synchronization purposes, the time indicated by the wall clock increases at real time speed (1 second per second), regardless of the duration of content that has been presented by the DASH client.

It is critical to synchronize the clocks of the DASH client and service when using a dynamic presentation because the MPD timeline of a dynamic presentation is mapped to wall clock time and many playback decisions are clock driven and assume a common understanding of time by the DASH client and service.

The time indicated by the wall clock does not necessarily need to match some universal standard as long as the DASH client and service are mutually synchronized.

Clock synchronization mechanisms are described by `UTCTiming` elements in the MPD ([DASH] 5.8.4.11).

The MPD of a dynamic presentation 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 [DASH].

A client presenting a dynamic presentation 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.

A DASH client SHALL NOT use a synchronization method that is not listed in the MPD unless explicitly instructed to do so by the application developer.

The use of a "default time source" by DASH clients is not allowed because this often obscures interoperability problems and introduces inconsistent behavior due to device clock differences.

The set of time synchronization mechanisms SHALL be restricted to the following subset of schemes from among those defined in [DASH] 5.8.5.7:

- `urn:mpeg:dash:utc:http-xsdate:2014`
- `urn:mpeg:dash:utc:http-iso:2014`
- `urn:mpeg:dash:utc:http-head:2014`
- `urn:mpeg:dash:utc:direct:2014`

ISSUE 1 We could benefit from some detailed examples here, especially as clock sync is such a critical element of live services. ¶

§ 13.2. Leap seconds

Dynamic presentations that cross the boundary between December/January or June/July ([LEAP-SECONDS]) need to correctly represent the effects of leap seconds to DASH clients, which shift the MPD timeline start point. Under the model defined by [DASH] 5.13, clients are informed of necessary leap second adjustments via the MPD.

The MPD of a dynamic presentations SHALL publish leap second offset information in the MPD, in the form of a `LeapSecondInformation` element as defined by [DASH] 5.13, unless the service provider does not intend for the presentation to remain accessible long enough to encounter a December/January or June/July transition in the UTC timezone.

Clients SHALL process leap second offset information (and any updates received due to MPD refreshes) in order to accurately calculate the effective availability start time.

§ 13.3. 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 ([DASH] 3.1.6). During playback of a dynamic presentation, new media segments continuously become available and stop being available with the passage of time.

An ***availability window*** is a time span on the MPD timeline that determines which media segments clients can expect to be available.

Each adaptation set has its own availability window. Services SHALL NOT define MPD attributes that affect the availability window on the representation level.

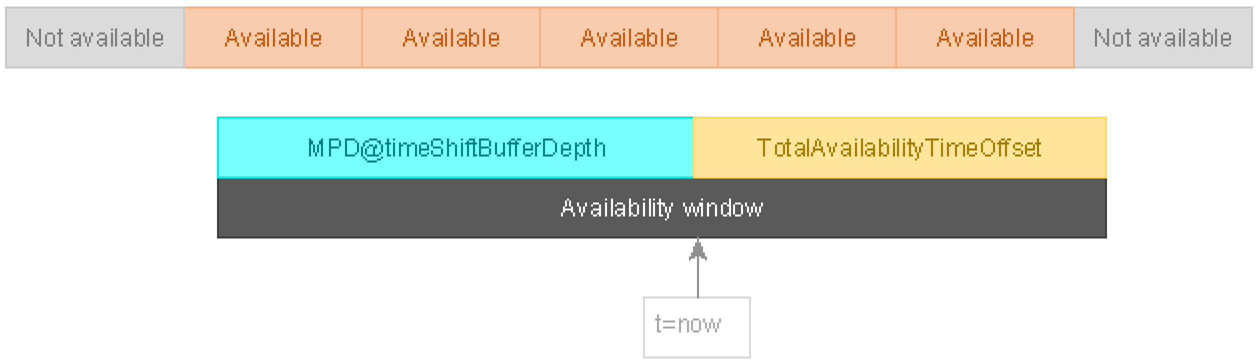

Figure 14 *The availability window determines which media segments can be expected to be available, based on where their segment end point lies.*

NOTE: A DASH service will typically make media segments available some seconds ahead of `t=now`, depending on its configuration and latency target. Furthermore, some periods may be entirely prepared in advance and available at all times (e.g. ads inserted between truly live content).

The availability window is calculated as follows:

1. Let *now* be the current wall clock time according to the wall clock.
2. Let *AvailabilityWindowStart* be `now - MPD@timeShiftBufferDepth`.
    - If `MPD@timeShiftBufferDepth` is not defined, let *AvailabilityWindowStart* be the effective availability start time.
3. Let *TotalAvailabilityTimeOffset* be the sum of all `@availabilityTimeOffset` values that apply to the adaptation set, either via `SegmentBase`, `SegmentTemplate` or `BaseURL` elements ([DASH] 5.3.9.5.3).
4. The availability window is the time span from *AvailabilityWindowStart* to `now + TotalAvailabilityTimeOffset`.

Media segments that have their segment end point inside or at the end of the availability window are available [DASH].

NOTE: [DASH] 4.3 and 5.3.9 define "segment availability time" of a segment as the span of wall clock time during which that media segment is available. Consequently, the availability window at each moment is **approximately** equivalent to the union of "segment availability times" of all available media segments at that moment.

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

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.

Despite best efforts, DASH services occasionally fail to achieve the availability windows advertised in the MPD. To ensure robust behavior even in the face of imperfect services, 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.

§ 13.4. 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 wall clock (`now`).

This is the mechanism by which clients can introduce a ***time shift*** (an offset) between wall clock time and the MPD timeline when presenting dynamic presentations. The time shift is zero when a client is presenting the media segment at the end point of the time shift buffer. By playing back media segments from further in the past, a positive 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. These factors often force a client to introduce a time shift:

1. § 13.3 Availability - not every media segment in the time shift buffer is guaranteed to be available.
2. § 13.5 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 the effective availability start time.

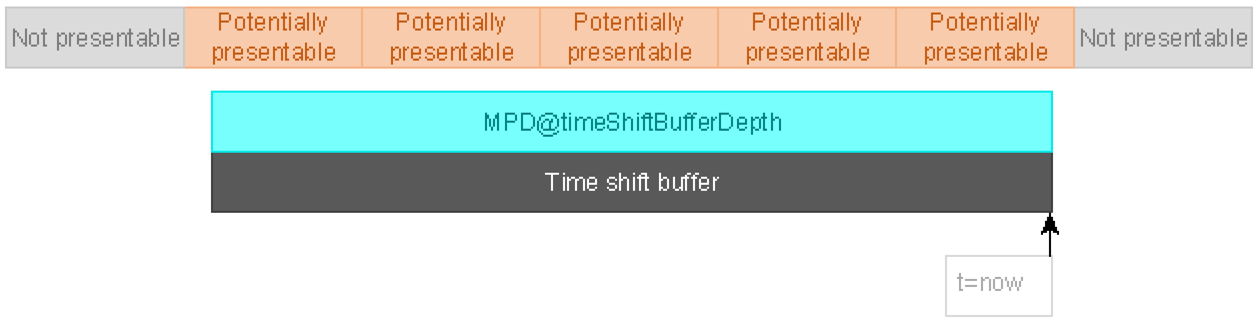

***Figure 15** Media segments overlapping the time shift buffer may potentially be presented by a client if other constraints do not forbid it.*

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. Common reasons for this are:

- The presentation just started and there is not enough content to fill the time shift buffer.
- The service is published with an effectively infinite time shift buffer (up to the zero point of the MPD timeline as indicated by the effective availability start time).

A dynamic presentation 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.

Clients SHALL NOT allow seeking into regions of the time shift buffer that are not covered by periods, regardless of whether such regions are before or after the periods described by the MPD.

§ 13.5. 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.

The mechanism that allows DASH clients to resolve this conflict is the ***presentation delay***, which decreases the time shift buffer by moving its end point into the past, creating an effective time shift buffer with a reduced duration.

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.

ISSUE 2 Can we recommend some meaningful algorithm for this? Something to use as a starting point would be nice to provide. ¶

The information required to calculate an optimal presentation delay might not always be available to DASH clients (e.g. because the client is not yet aware of upcoming periods that will be added to the MPD later and will significantly change the optimal presentation delay). Services MAY define the `MPD@suggestedPresentationDelay` attribute to provide a suggested presentation delay. Clients SHOULD use `MPD@suggestedPresentationDelay` when provided by the MPD, ignoring any 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.

A common error in DASH content authoring is to attempt to use `MPD@minBufferTime` to control the presentation delay. `MPD@minBufferTime` is not related to presentation delay and merely describes the allowed jitter in content encoding ([DASH] 5.3.5.2), as determined by the encoder or derived from the encoder configuration.

The ***effective time shift buffer*** is the time span from the start of the time shift buffer to `now - PresentationDelay`. Services SHALL NOT define a value for `MPD@suggestedPresentationDelay` that results in an effective time shift buffer of negative or zero duration.

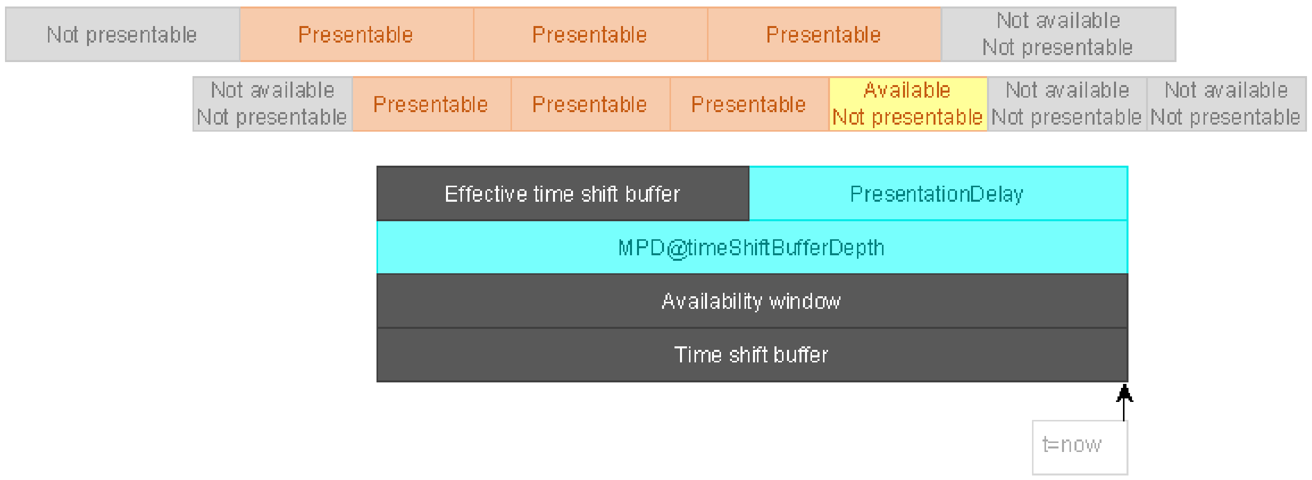

Figure 16 *Media segments that overlap the effective time shift buffer are the ones that may be presented at time* `now`*. Two representations with different segment lengths are shown. Diagram assumes* `@availabiltiyTimeOffset=0`*.*

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.

§ 13.6. MPD updates

The MPD of a dynamic presentation 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 the MPDs of dynamic presentations:

- 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 editorial changes in content scheduling take place.
- Removing `MPD@minimumUpdatePeriod` to signal that MPD will no longer be updated.
- Converting the presentation to a static presentation to signal that a live service has become available on-demand as a recording.

[DASH] 5.4.1 defines various constraints for MPD updates, most importantly:

- `MPD@id` does not change.
- `MPD.Location` does not change.
- `Period@id` 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.

DASH-IF implementation guidelines further extend these constraints:

- `MPD@availabilityStartTime` SHALL NOT change.
    - Leap second adjustments are performed by adjusting the `LeapSecondInformation` element (see § 13.2 Leap seconds).
- `Period@start` SHALL NOT change.
- `Period@duration` SHALL NOT change except when explicitly allowed by other statements in this document.
- `AdaptationSet@id` SHALL be present on every `AdaptationSet` element.
- The set of adaptation sets present in an existing period (i.e. the set of `AdaptationSet@id` values) SHALL NOT change.
- The relative order of `AdaptationSet` elements in a `Period` element SHALL NOT change.
- The representations present in an existing adaptation set (i.e. the set of `Representation@id` values) SHALL NOT change.
- The relative order of `Representation` elements in an `AdaptationSet` element SHALL NOT change.

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

Clients SHALL use `@id` to track period, adaptation set and representation identity across MPD updates (instead of relying on, for example, the order of XML elements).

It will take some time for each MPD update to reach clients, both due to the MPD validity duration and network connectivity influences. The constraints in this document set some limits on the data allowed to change with an MPD update in order to prevent changes in data already processed by DASH clients. Services SHOULD perform changes well in advance of the changed data being processed by clients.

§ 13.6.1. MPD snapshot validity

The MPD of a dynamic presentation remains valid not only at its moment of initial publishing but through the entire ***MPD validity duration***, which is a time span of duration `MPD@minimumUpdatePeriod` starting from the moment the MPD download is started by a client ([DASH] 5.4.1).

Validity means that the MPD remains in conformance to all requirements defined by [DASH] and this document. For example, any MPD of a dynamic presentation must include enough segment references to cover a time span of `MPD@minimumUpdatePeriod` into the future, in addition to the segment references that would ordinarily be expected at time of initial download. See also § 9.2.2 Necessary segment references in dynamic presentations.

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 downloading a new snapshot of the MPD.

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

The presence or absence of `MPD@minimumUpdatePeriod` SHALL be used by DASH services to signal whether and when the MPD might be updated (with presence indicating potential for future updates):

- A nonzero value for `MPD@minimumUpdatePeriod` defines the MPD validity duration of the present snapshot of the MPD, starting from the moment its download was initiated. This allows the service to provide regular updates to the MPD while limiting the refresh interval to avoid overload.
- The value 0 for `MPD@minimumUpdatePeriod` indicates that the MPD has no validity after the moment it is retrieved. In such a situation, the client SHALL acquire a new MPD whenever it wants to make new media segments available (no "natural" state changes will occur due to passage of time).

- Absence of the `MPD@minimumUpdatePeriod` attribute indicates an infinite validity (the MPD will never be updated).
    - One typical use case is to combine this with an infinite sequence of segment references, defining a presentation that never ends and never changes.
    - Another use case is using a dynamic presentation to schedule the presentation of pre-prepared finite content for a specific time span of wall clock time.

§ *13.6.1.1. In-band MPD validity events*

In addition to the `MPD@minimumUpdatePeriod` mechanism for defining the MPD validity duration, a DASH service MAY publish in-band MPD validity update events ([DASH] 5.10.4.2). If a DASH client processes in-band events for determining the MPD snapshot validity duration then `MPD@minimumUpdatePeriod` is ignored for the purposes of determining MPD snapshot validity.

When in-band signaling is used, the absence of an in-band event that corresponds to a particular MPD snapshot (identified by `MPD@publishTime`) implies MPD snapshot validity extension until an explicit validity duration is defined by a future in-band event. This enables finer control over MPD snapshot validity by the service but might not be supported by all clients.

> NOTE: Effectively, there are two MPD snapshot validity durations in place with in-band signaling, one defined by `MPD@minimumUpdatePeriod` and one by in-band signaling. A DASH client may use either. DASH services are sometimes published with `MPD@minimumUpdatePeriod=0` in such a situation, reducing the validity duration defined by one model to zero and allowing the other model to have full control. This may cause extra overhead for clients that do not use in-band signals, however.

Services SHALL NOT require clients to support in-band events - it is an optional optimization mechanism to allow clients to reduce HTTP traffic caused by fetching new MPD snapshots.

§ 13.6.2. Adding content to the MPD

[DASH] allows the following 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.

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. An MPD update that adds content MAY be combined with an MPD update that removes content.

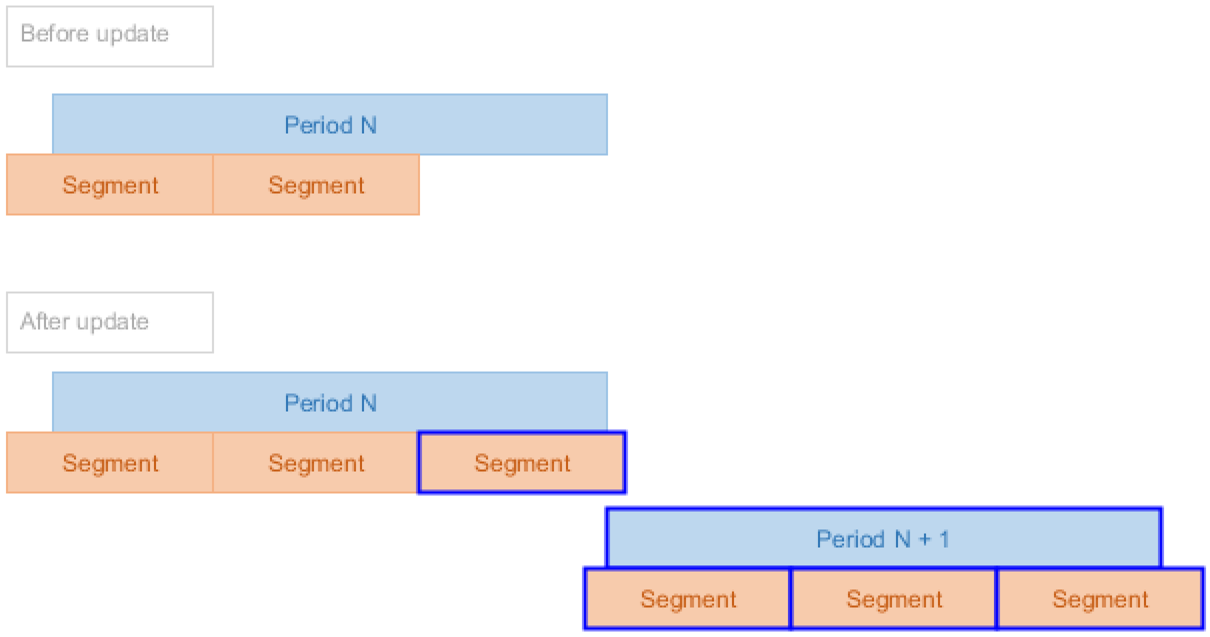

***Figure 17** MPD updates can add both segment references and periods (additions highlighted in blue).*

The duration of the last period cannot change as a result of adding segment references. A live service will typically 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 `S@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.

§ 13.6.3. Removing content from the MPD

[DASH] allows the following mechanisms for 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 timeline.
- Expired periods and segment references that no longer overlap the time shift buffer may be removed from the start of the MPD timeline.

Multiple content removal mechanisms MAY be combined in a single MPD update. An MPD update that removes content MAY be combined with an MPD update that adds content.

Removal of content is only allowed if the content to be removed is expired or not yet available to clients and guaranteed not to become available within the MPD validity duration of any MPD snapshot potentially downloaded by clients.

To determine the content that may be removed, calculate `EarliestRemovalPoint` as follows for each adaptation set:

1. Let *PublishingDelay* be the end-to-end delay for MPD update publishing. This is the time between the MPD generator creating a new version of the MPD and this new version becoming published to all clients on the CDN edge (but not necessarily downloaded yet by any of them).
2. Let *AvailabilityWindowEnd* be the end point of the availability window.
3. Let `EarliestRemovalPoint` be *AvailabilityWindowEnd* + `MPD@minimumUpdatePeriod` + *PublishingDelay*.

NOTE: As each adaptation set has its own availability window, so does each adaptation set have its own `EarliestRemovalPoint`.

Media segments that overlap or end before `EarliestRemovalPoint` might be considered by clients to be available at the time the MPD update is processed. Therefore, an MPD update removing content SHALL NOT remove any segment references to media segments with a segment start point before or at `EarliestRemovalPoint`.

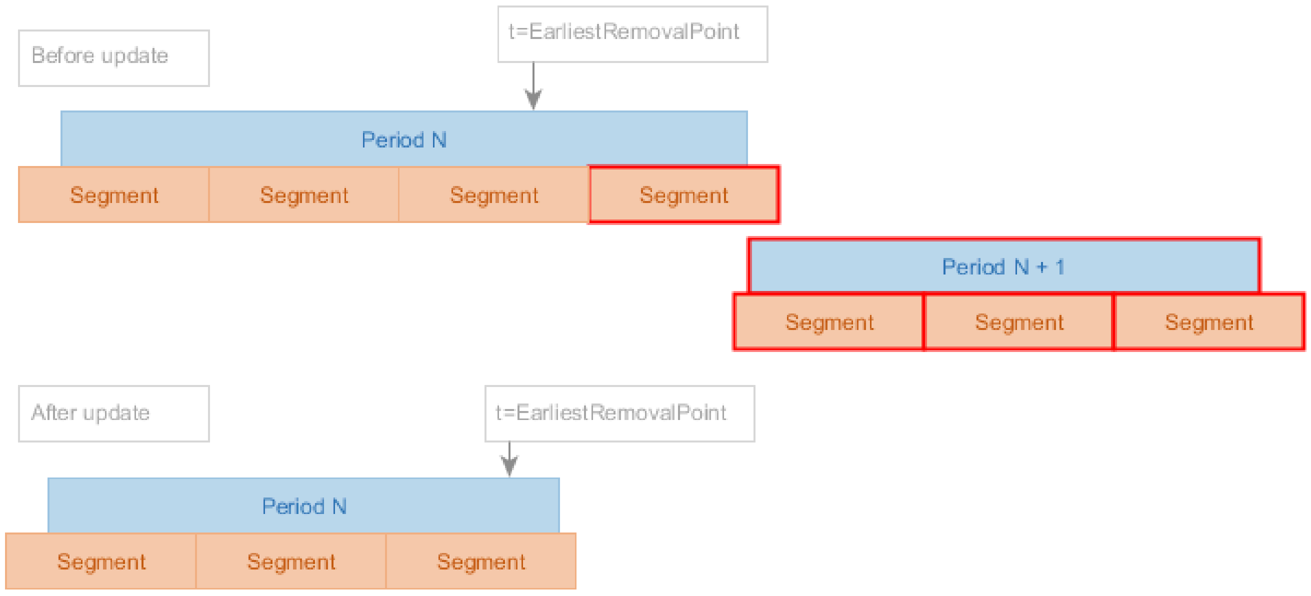

Figure 18 *MPD updates can remove both segment references and periods (removals highlighted in red).*

Perfect handling of `EarliestRemovalPoint` by every service cannot be assumed. 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 or at time of MPD update processing. 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 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.

The above logic implements the removal of expired unnecessary segment references according to the rules defined in § 9.2.3 Removal of unnecessary segment references.

§ 13.6.4. 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.

When an MPD is updated to a state that describes the final content of a live service, the service SHALL define a fixed duration for the last period, remove the `MPD@minimumUpdatePeriod` attribute and cease performing MPD updates to signal to clients that no more content will be added to the MPD.

Upon detecting the removal of `MPD@minimumUpdatePeriod` clients SHOULD present a user experience suitable for end of live content.

A common mistake is to treat the eventual cessation of new content as a transient or fatal error (resulting in potentially infinite loading even before the final media segment is presented to the user).

If the ending live service is to be converted to a static presentation for on-demand viewing, the service MAY change `MPD@type` to `static` when `MPD@minimumUpdatePeriod` is removed or do so at a later time. The resulting static presentation MAY remain accessible on the same URL as the original dynamic presentation. For content changes performed simultaneously with the `MPD@type` change, the same rules apply as for regular MPD updates of dynamic presentations.

NOTE: The MPD update constraints mean that making a live service available for on-demand viewing on the same URL, by transforming `MPD@type` to `static`, does not allow for historical (expired) content to once again become available. To enable expired content to be published on-demand, the MPD should be published on a new URL as an independent static presentation.

Clients SHALL NOT lose track of the playback position if a dynamic presentation becomes a static presentation, even if the time span of the static presentation exceeds the time shift buffer of the dynamic presentation (e.g. because the static presentation includes content further into the future).

The MPD of a dynamic presentation in which all content has expired (and which is not converted to a static presentation for on-demand viewing) SHOULD be unpublished, resulting in a 404 Not Found status when clients attempt to access the MPD.

§ 13.7. MPD refreshes

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

Clients presenting dynamic presentations SHALL execute the following MPD refresh logic:

1. When an MPD snapshot is downloaded, it is valid for the MPD validity duration as measured from the moment the download is initiated. See §13.6.1 MPD snapshot validity.
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 MPD validity duration is extended to `DownloadStart + 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.

§ 13.7.1. Conditional MPD downloads

It can often be the case that a live service signals a short MPD validity duration 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.

§ 14. Segment loss handling

Due to network or other faults, it is possible that media segments do not reach the DASH packager, effectively creating a discontinuity in a representation. As DASH clients typically have difficulties processing content with gaps and the timing model described here forbids gaps in general, missing segments would likely lead to an unsatisfactory playback experience for end-users.

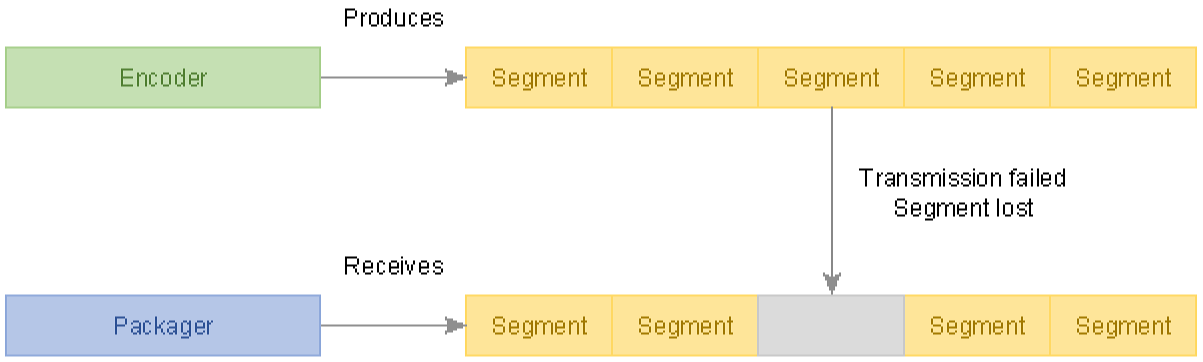

Figure 19 *A DASH packager might not have every media segment available when it needs to publish them. Corrective actions must be taken to ensure an uninterrupted timeline is presented to DASH clients.*

Therefore, DASH services SHALL NOT publish periods that have missing segments, whether the segment loss is described by "missing content segments" ([DASH] 6.2.6) or by any other means (including not describing it).

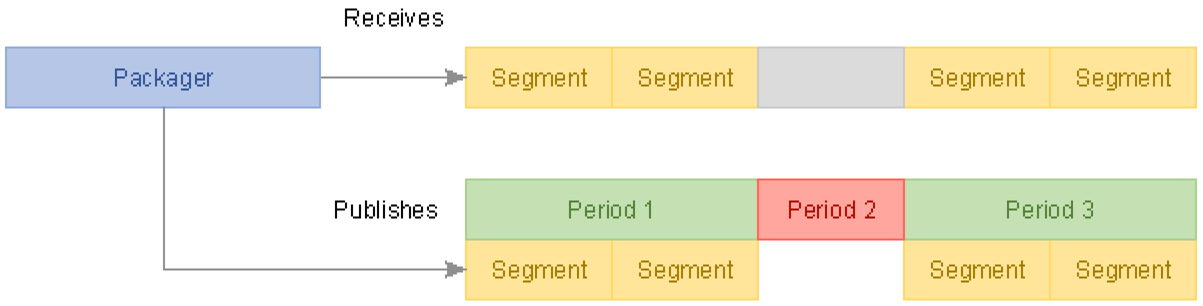

Figure 20 *The simplest correction is to start a new period that does not include the affected representation for the duration of the loss. Other representations remain present and a client can often continue seamless playback without the missing representation.*

Instead, DASH services SHOULD start a new period that does not include the representation that would experience a gap, later restoring the representation with a new period transition. Period-connected adptation sets can enable DASH clients to perform such transitions seamlessly in some scenarios.

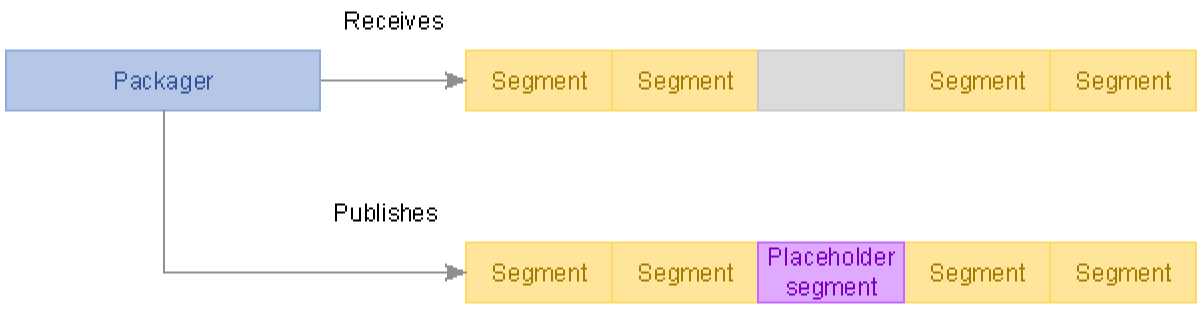

Figure 21 *Other solutions might involve replacing the missing media segment with a placeholder, either from a different representation or an entirely artificial one.*

Some DASH clients experience difficulties when transitioning to/from a very short period (e.g. with a duration of only 1 media segment). Implementations MAY extend the transition period for better compatibility with such clients.

Alternatively, given a sufficiently capable DASH packager and provided that technical constraints of representations are satisfied, the missing media segment MAY be replaced with an aligned media segment from a lower bitrate (likely requires a single initialization CMAF switching set [CMAF] 7.3.4.2).

§ 15. 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 [ISOBMFF] encapsulation.

NOTE: Storing text tracks in stand-alone files is not permitted by [CMAF]. If you intend your DASH service to conform to [CMAF], you must store text tracks as segmented [CMAF] tracks.

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 6
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.

§ 16. Forbidden techniques

Some aspects of [DASH] 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. This information serves no purpose under the interoperable timing model.
- The `@availabilityTimeComplete` attribute SHALL NOT be used. The concept of "incomplete but available" media segments that this attribute enables is not part of the interoperable timing model.
- There SHALL NOT be "missing content segments" ([DASH] 6.2.6) in the content. If content is lost during processing, the expectation is that the encoder/packager will either replace it with valid content (e.g. content from a lower representation or potentially even blank picture or silent audio) or start a new period that does not contain the representation that incurs data loss.

§ 17. Examples

This section is informative.

§ 17.1. 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.

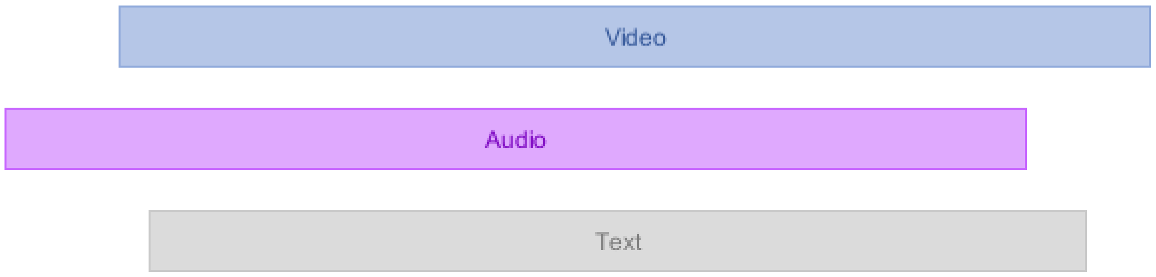

Figure 22 *Content with different track lengths, before packaging as DASH.*

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.

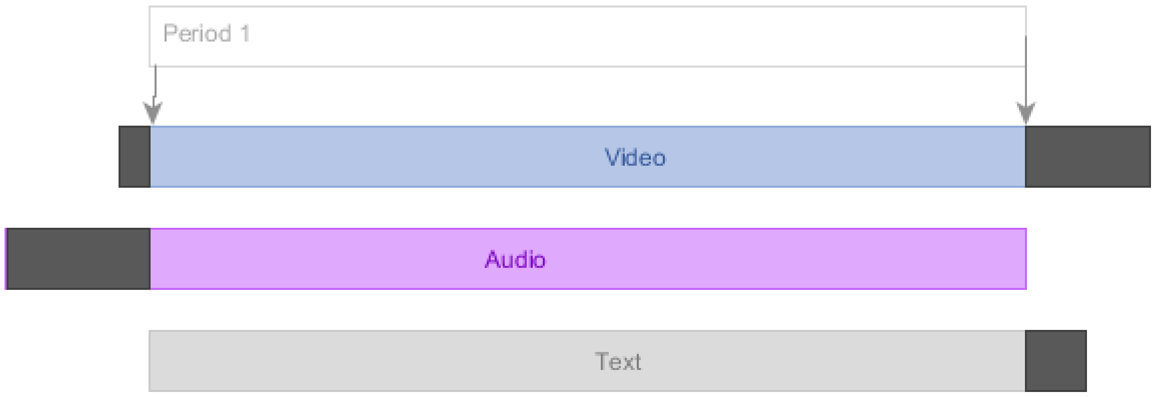

Figure 23 *Content may be cut (indicated in black) to equalize track lengths.*

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.

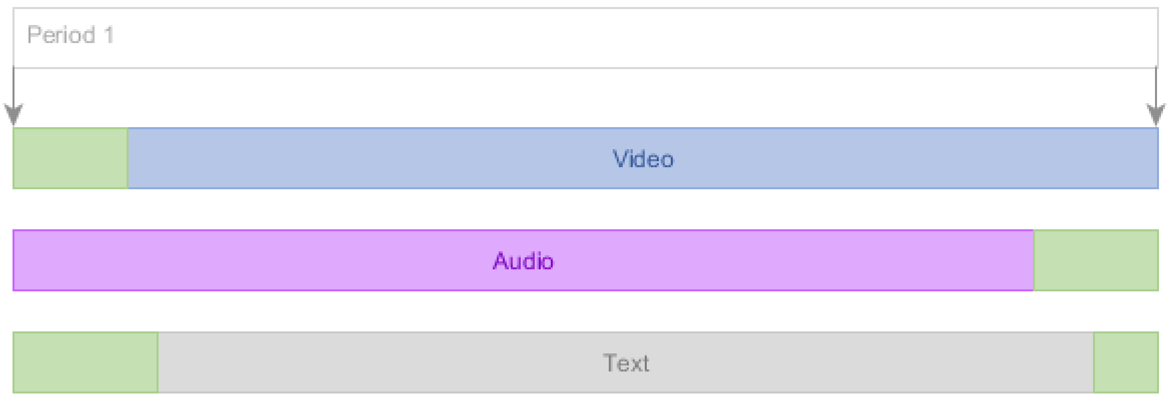

Figure 24 *Content may be padded (indicated in green) to equalize track lengths.*

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.

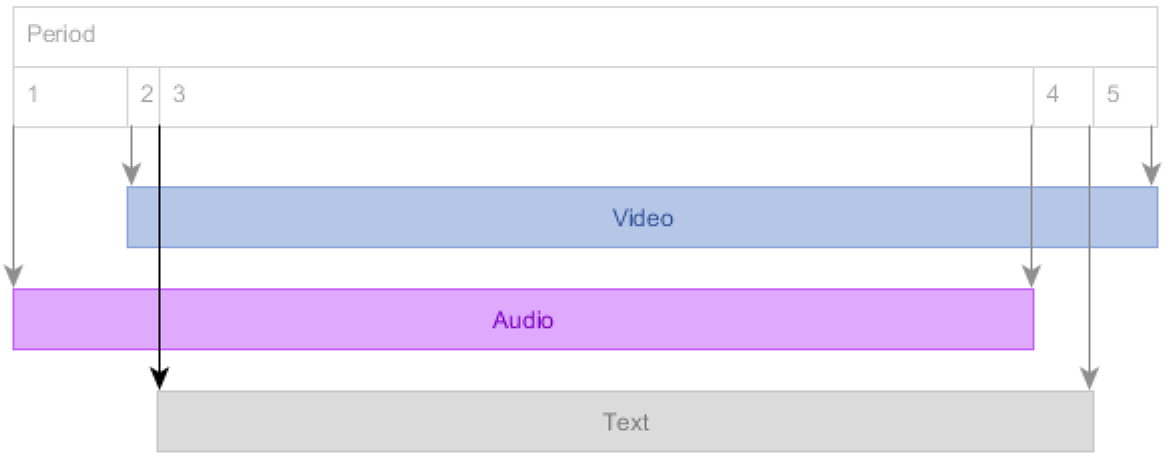

Figure 25 *New periods may be started at any change in the set of available tracks.*

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 26 *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.

§ 17.2. 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.

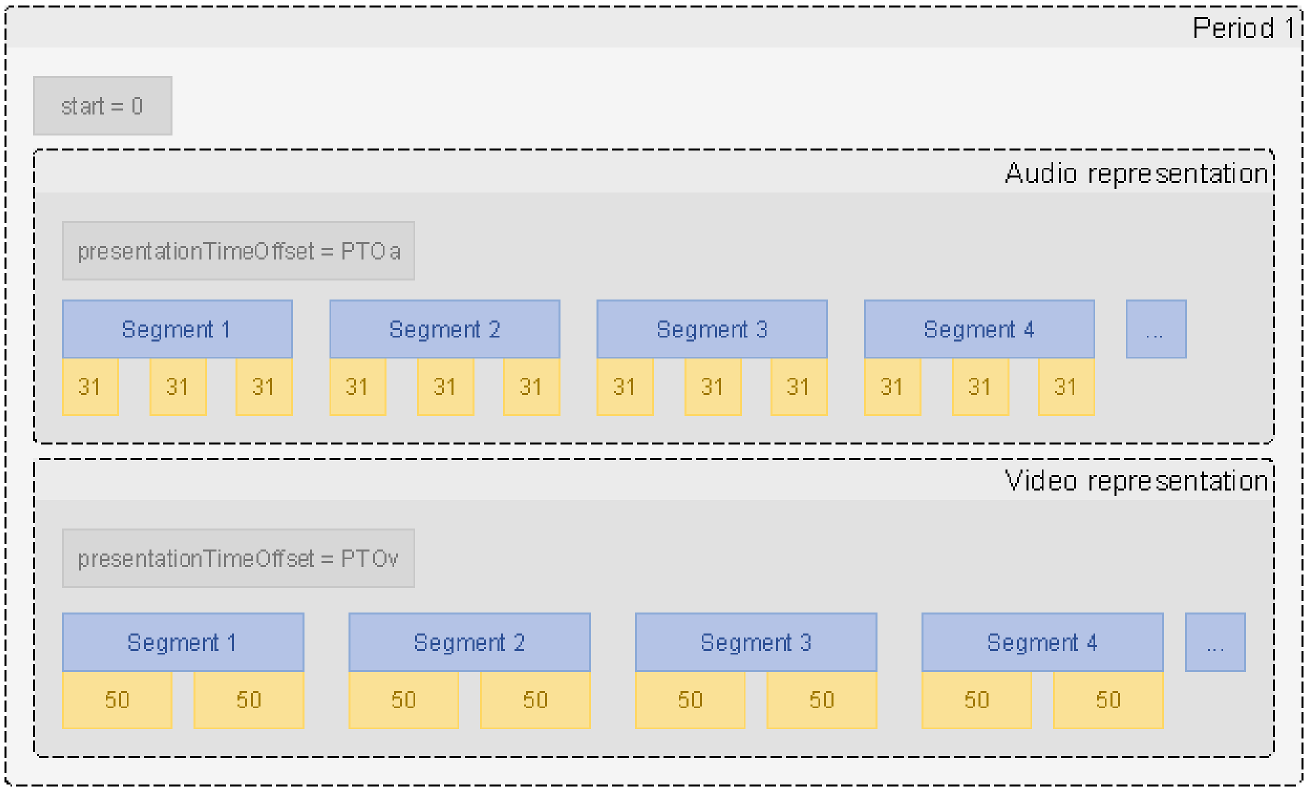

Figure 27 *Presentation with one period, before splitting. Blue is a segment, yellow is a sample. Duration in arbitrary units is listed on samples. Segment durations are taken to be the sum of sample durations.* `presentationTimeOffset` *may have any value - it is listed because will be referenced later.*

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 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.

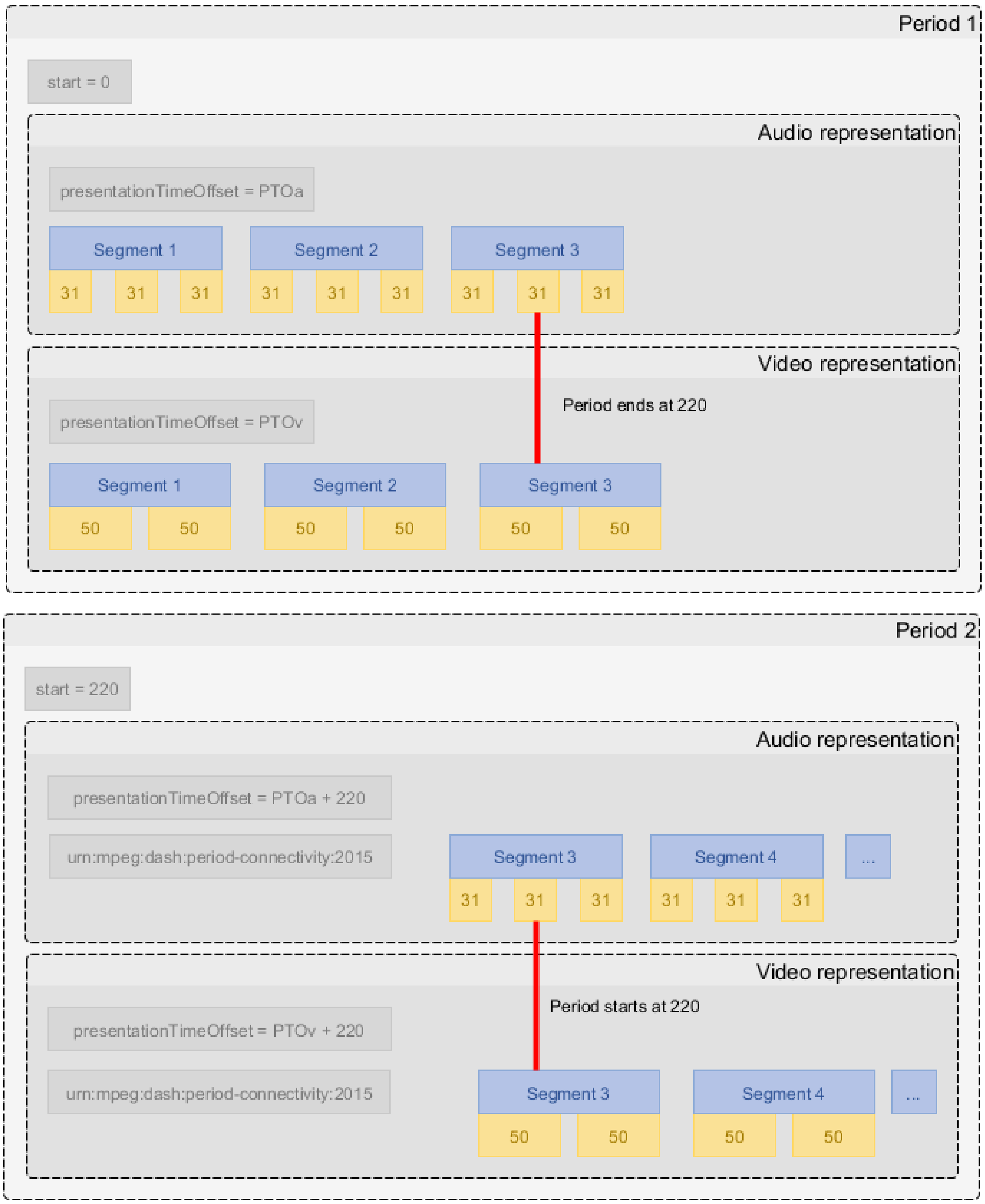

Figure 28 *Presentation with two periods, after splitting. Audio segment 3 and video segment 3 are shared by both periods, with the connectivity signaling indicating that seamless playback with de-duplicating behavior is expected from clients.*

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 typically unsuitable for some multi-period scenarios. See § 18.4.2 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.

§ 17.3. 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.

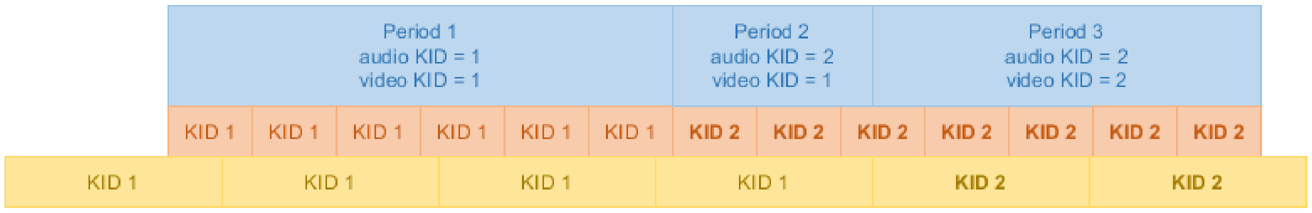

Figure 29 *A change in `default_KID` starts a new period. Orange indicates audio and yellow video representation.*

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

§ 18. Segment addressing modes

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

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.

§ 18.1. 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: [DASH] 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 [CMAF].

Figure 30 *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 ([DASH] 5.3.9.2). 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 `tkhd` box [ISOBMFF]).

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 7
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="848-99
          <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.

18.2. Structure of the index segment

The index segment SHALL consist of a single Segment Index Box (`sidx`) as defined by [ISOBMFF]. The field layout of this data structure 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 SHOULD be determined as follows:

NOTE: The normative definitions of the fields are provided by [ISOBMFF]. This document describes how to determine the correct values, relating the fields to DASH specific concepts.

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 3 We need to clarify how to determine the right value for `SAP_type` GitHub #235. ¶

§ 18.2.1. 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.

§ 18.3. 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 sample timeline time span covered by the samples in 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.

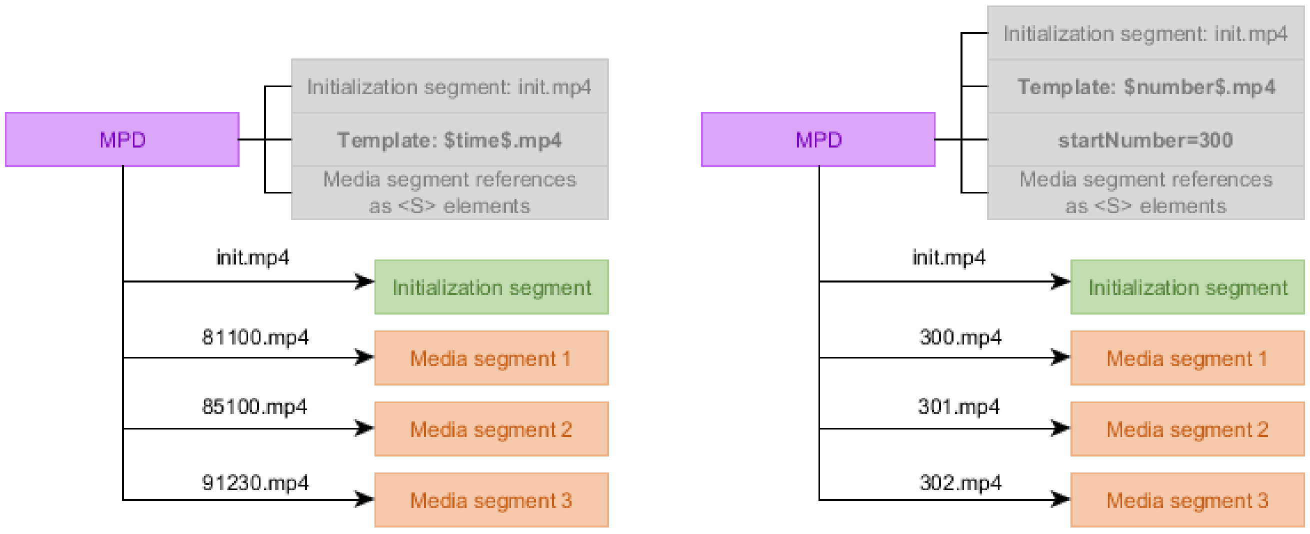

***Figure 31** Explicit addressing uses a segment template that is combined with explicitly defined time spans for each media segment in order to reference media segments, either by start time or by sequence number.*

The MPD SHALL contain a `SegmentTemplate/SegmentTimeline` element, containing a set of segment references that 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 ([DASH] 5.3.9.6). The `SegmentTemplate@duration` attribute is not present ([DASH] 5.3.9.2).

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` indicates the number of additional consecutive media segments that exist ([DASH] 5.3.9.6).

NOTE: Only additional segment references are counted by `@r`, so `S@r=5` indicates a total of 6 consecutive media segments with the same duration.

The segment start point is calculated by adding the segment start point and duration of the previous media segment, unless `S@t` is specified in which case `S@t` is the segment start point on the sample timeline ([DASH] 5.3.9.6).

The value of `S@r` is nonnegative, except for the last `S` element which MAY have a negative value in `S@r` ([DASH] 5.3.9.6), 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 the MPD of a dynamic presentation 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 `S@n` attribute SHALL NOT be used - segment numbers form a continuous sequence starting with `SegmentTemplate@startNumber`.

The `SegmentTemplate@eptDelta` attribute SHALL NOT be present. The information represented by this attribute can be calculated independently and having it be present would only create additional possibility for conflicting data.

The `SegmentTemplate@media` attribute SHALL contain the URL template for referencing media segments. The `SegmentTemplate@initialization` attribute SHALL contain the URL template for referencing initialization segments.

Either the `$Time$` or `$Number$` template variable SHALL be present in `SegmentTemplate@media` to uniquely identify media segments:

- If using `$Number$` addressing, the number of the first segment reference is defined by `SegmentTemplate@startNumber` (default value 1) ([DASH] 5.3.9.5.3).
- If using `$Time$` addressing, the value for each segment reference is the segment start point on the sample timeline, in timescale units ([DASH] 5.3.9.5.3).

EXAMPLE 8

Below is an example of common usage of explicit addressing.

The example defines 225 media segments starting at position 900 (or 0.9 seconds) on the sample timeline and lasting for a total of 900.225 seconds. The period is only 900 seconds long, 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 start point is at position 900 (or 0.9 seconds) on the sample timeline 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 9

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 (or 0.12 seconds) 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 start point on the sample timeline is at position 810 (or 0.81 seconds), 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.

§ 18.3.1. 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 § 9 Representation timing and § 13.6.3 Removing content from the MPD to understand the constraints that apply to segment reference removal.

§ 18.4. 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 MPD describing the nominal time span of the sample timeline covered by each media segment.

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 § 18.4.1 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 32__ 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. Note that `@eptDelta` does not affect the generated paths!

The `SegmentTemplate@duration` attribute defines the nominal duration of a media segment in timescale units ([DASH] 5.3.9.2).

The set of segment references consists of the first media segment starting `SegmentTemplate@eptDelta` timescale units relative to 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 `@eptDelta` attribute SHALL be present if its value is not zero.

NOTE: `@eptDelta` is expressed as an offset from the period start point to the segment start point of the first media segment ([DASH] 5.3.9.2). In other words, the value will be negative if the first media segment starts before the period start point.

`@eptDelta` is new in [DASH] 4th edition (published 2020) and DASH client support is not yet widespread. Clients that do not implement support for `@eptDelta` may fail to correctly begin or end playback of periods that use simple addressing with `@eptDelta != 0`. If the client cannot be upgraded to consider `@eptDelta` then you are advised to use explicit addressing with such content.

The `SegmentTemplate@media` attribute SHALL contain the URL template for referencing media segments. The `SegmentTemplate@initialization` attribute SHALL contain the URL template for referencing initialization segments.

Either the `$Time$` or `$Number$` template variable SHALL be present in `SegmentTemplate@media` to uniquely identify media segments:

- If using `$Number$` addressing, the number of the first segment reference is defined by `SegmentTemplate@startNumber` (default value 1) ([DASH] 5.3.9.5.3).
- If using `$Time$` addressing, the template value for each segment reference is the segment start point on the sample timeline minus `@eptDelta` ([DASH] 5.3.9.5.3).

EXAMPLE 10 ¶

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 (or 0.9 seconds) on the sample timeline and the first media segment starting at position 400 (or 0.4 seconds). The average duration of a media segment is 4001 (4.001 seconds). 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 226 defined segment references.

The period start point is 500 milliseconds after the segment start point of the first media segment and the period end point is approximately 275 milliseconds after the segment start point of the last media segment. The real timing of the samples within the media segments may differ from these nominal values in the MPD, to the extent permitted by the timing model.

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011">
  <Period duration="PT900S">
    <AdaptationSet>
      <Representation>
        <SegmentTemplate timescale="1000" presentationTimeOffset="900" eptDelta="-500"
            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.

§ 18.4.1. 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 33** 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.

The maximum deviation of either edge is 50% of the nominal media segment duration and MAY be in either direction ([DASH] 7.2.1).

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 contraint 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).

To ensure that no gaps in the timeline are introduced by the allowed inaccuracy, additional constraints apply to the contents of media segments at the edges of a period:

- The [=media segment] that starts at or overlaps the period start point SHALL contain a sample that starts at or overlaps the period start point.

- The [=media segment] that ends at or overlaps the period end point SHALL contain a sample that ends at or overlaps the period end point.

EXAMPLE 11 ¶

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 nonconforming because of the requirement that the first media segment contain a media sample that starts at or overlaps the period start point.

§ 18.4.2. 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. This can be challenging when using simple addressing.

The media segment that overlaps the period start point must contain a sample that starts at or overlaps the period start point. Likewise, the media segment that overlaps the period end point must contain a sample that ends at or overlaps the period end point. These constraints are defined in § 18.4.1 Inaccuracy in media segment timing when using simple addressing and typically make it impossible to move the period start point or split a period when using simple addressing and taking advantage of the inaccuracy allowed to exist between nominal timing of the sample timeline and the true contents of the media segments.

The rest of this chapter assumes that the nominal timing of media segments matches the real timing. If you cannot satisfy this constraint but still wish to move the period start point, convert to explicit addressing. See § 18.4.3 Converting simple addressing to explicit addressing.

To move the period start point for representations that use simple addressing without timing inaccuracy:

1. Update `SegmentTemplate@presentationTimeOffset` to indicate the desired period start point on the sample timeline.
2. Update `SegmentTemplate@eptDelta` to indicate the relative position of the segment start point of the first media segment from the start of the period (with a negative sign indicating the segment start point is before the period start point).
3. If using the `$Time$` template variable and if the value of `@eptDelta` changed in the previous step, rename all media segments to conform to the new pattern generated by the URL template. The pattern will change whenever `@eptDelta` changes because `$Time$` refers not only to the segment start point but also includes `@eptDelta`.
4. If using the `$Number$` template variable, increment `SegmentTemplate@startNumber` by the number of media segments removed from the beginning of the representation.
5. Update `Period@duration` to match the new duration.

`@eptDelta` is new in [DASH] 4th edition (published 2020). If the resulting `SegmentTemplate@eptDelta` value is not zero, DASH clients that do not support `@eptDelta` may exhibit incorrect behavior when transitioning between periods. The only workaround is to either convert to explicit addressing or to choose a period start point that overlaps with the segment start points of all representations in all adaptation sets that use simple addressing! Such points might not exist, depending on the media segment structure of the presentation.

18.4.3. 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 the MPD.

To perform the conversion, execute the following steps:

1. Calculate the number of media segments in the representation as `SegmentCount = Ceil((AsSeconds(Period@duration) - AsSeconds(SegmentTempalte@eptDelta)) / ( 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` plus `@eptDelta`.
    4. Set `S@d` to equal `SegmentTemplate@duration`.
    5. Remove `SegmentTemplate@duration`.
    6. Set `S@r` to `SegmentCount - 1`.
    7. Remove `SegmentTemplate@eptDelta`. It is not needed nor permitted with explicit addressing.

3. If using `$Time$` addressing in `SegmentTemplate@media`, rename all media segments to match the segment start point in the template variable (simple addressing uses segment start point minus `@eptDelta` for `$Time$`).

EXAMPLE 12

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" eptDelta="-500"
            media="video/$Number$.m4s" initialization="video/init.mp4"
            duration="4001" startNumber="800" />
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

As part of the conversion, we calculate `SegmentCount = Ceil((900 - (-0.5)) / (4001 / 1000)) = 226`.

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="400" 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.

§ 19. Large timescales and time values

[ECMASCRIPT] is unable to accurately represent numeric values greater than 2^{53} (`9007199254740991`) 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}.

EXAMPLE 13 ¶

The issue does not arise with the common 90 KHz timescale. Counting time since the Unix epoch until 11 November 2019 we get `141721093260000` which is well within the allowed range of values.

Another common timescale is 10000000 (10 million timescale units per second) often used by Smooth Streaming. Counting time since the Unix epoch until 11 November 2019 we get `15746788140000000` which does exceed the critical value and will result in broken playback on many clients! To correct such an error, use a smaller timescale or a MPD timeline zero point that is not so far in the past.

§ 20. 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.

§ Index

§ Terms defined by this specification

§ References

§ Normative References

[CMAF]
Information technology — Multimedia application format (MPEG-A) — Part 19: Common media application format (CMAF) for segmented media. February 2024. Published. URL: https://www.iso.org/standard/85623.html

[DASH]
Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats. Under development. URL: https://www.iso.org/standard/89027.html

[DASH-CMAF]
Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats — Amendment 1: Media presentation insertion event, nonlinear playback and other extensions. Deleted. URL: https://www.iso.org/standard/83315.html

[ISOBMFF]
Information technology — Coding of audio-visual objects — Part 12: ISO base media file format. Under development. URL: https://www.iso.org/standard/85596.html

[MPEG2TS]
Information technology — Generic coding of moving pictures and associated audio information — Part 1: Systems. December 2023. Published. URL: https://www.iso.org/standard/87619.html

[RFC2119]
S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. March 1997. Best Current Practice. URL: https://datatracker.ietf.org/doc/html/rfc2119

[RFC7232]
R. Fielding, Ed.; J. Reschke, Ed.. *Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests*. June 2014. Proposed Standard. URL: https://httpwg.org/specs/rfc7232.html

[RFC7233]
R. Fielding, Ed.; M. Nottingham, Ed.; J. Reschke, Ed.. *HTTP Semantics*. June 2022. Internet Standard. URL: https://httpwg.org/specs/rfc9110.html

§ Informative References

[ATSC3]
ATSC Standard: A/300:2017 "ATSC3.0 System". URL: https://https://www.atsc.org/wp-content/uploads/2017/10/A300-2017-ATSC-3-System-Standard-1.pdf

[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

[ECMASCRIPT]
ECMAScript Language Specification. URL: https://tc39.es/ecma262/multipage/

[ENCRYPTED-MEDIA]
Joey Parrish; Greg Freedman. *Encrypted Media Extensions*. URL: https://w3c.github.io/encrypted-media/

[HLS]
R. Pantos, Ed.; W. May. *HTTP Live Streaming*. August 2017. Informational. URL: https://www.rfc-editor.org/rfc/rfc8216

[LEAP-SECONDS]
IERS Bulletin C (leap second announcements). URL: https://datacenter.iers.org/data/latestVersion/16_BULLETIN_C16.txt

[MEDIA-SOURCE]
Jean-Yves Avenard; Mark Watson. *Media Source Extensions™*. URL: https://w3c.github.io/media-source/

§ Issues Index

ISSUE 1 We could benefit from some detailed examples here, especially as clock sync is such a critical element of live services.

ISSUE 2 Can we recommend some meaningful algorithm for this? Something to use as a starting point would be nice to provide.

ISSUE 3 We need to clarify how to determine the right value for `SAP_type` GitHub #235.