TECHNICAL REPORT

ISO/IEC TR 23009-3

Second edition
2015-10-15

Information technology — Dynamic adaptive streaming over HTTP (DASH) —

Part 3: Implementation guidelines

Technologies de l'information — Diffusion en flux adaptatif dynamique sur HTTP (DASH) —

Partie 3: Lignes directrices de mise en oeuvre

Reference number
ISO/IEC TR 23009-3:2015(E)

Contents

Page

Foreword ..... **v**
Introduction ..... **vi**
1 Scope **1**
2 Normative references **1**
3 Terms, definitions, and abbreviated terms **1**
4 General **2**
4.1 System overview 2
4.2 Normative parts 2
4.3 Main design principles 3
4.3.1 Common timeline 3
4.3.2 Data model 3
4.3.3 Segments 4
4.3.4 Segment types 5
4.3.5 Segment addressing schemes 5
4.3.6 Stream access points 6
4.3.7 Remote elements 6
4.3.8 Events 7
4.3.9 General-purpose descriptors 7
4.4 Background on DASH profile concept 8
4.5 Dynamic aspects 8
5 Guidelines for content generation **10**
5.1 General guidelines 10
5.1.1 Video content generation 10
5.1.2 Audio content generation 12
5.1.3 Content preparation for live streaming 14
5.1.4 Guidelines for generation of segment file names 14
5.2 Guidelines for ISO-BMFF content generation 17
5.2.1 On-demand streaming 17
5.2.2 Live streaming 21
5.2.3 Enabling trick modes 23
5.2.4 Support for SubRepresentations 24
5.2.5 Enabling delivery format to storage file format conversion 26
5.3 Guidelines for MPEG-2 TS content generation 30
5.3.1 General recommendations 30
5.3.2 Live streaming 31
5.3.3 On demand streaming 32
5.4 Guidelines for Advertisement Insertion 33
5.4.1 Use cases 33
5.4.2 Architectures and workflows 34
5.4.3 App-driven ad insertion 36
5.5 DASH MPD and Segment-based Live Service Offering 37
5.5.1 Preliminaries 37
5.5.2 Service Offering Requirements and Guidelines 38
5.5.3 Client requirements and guidelines 41
5.6 Guidelines for low latency live service 43
5.6.1 Use case 43
5.6.2 General Approach: Chunked transfer 43
5.6.3 MPD generation 43
6 Client implementation guidelines **44**
6.1 General 44
6.2 Client architecture overview 44
6.3 Example of client operation 45

6.4 Timing model for live streaming 45
6.4.1 General 45
6.4.2 MPD information 45
6.4.3 MPD times 46
6.4.4 Context derivation 46
6.4.5 Derivation of MPD times 47
6.4.6 Addressing methods 47
6.4.7 Scheduling playout 48
6.4.8 Validity of MPD 48
6.5 MPD retrieval 48
6.6 Segment list generation 49
6.6.1 General 49
6.6.2 Template-based generation of segment list 50
6.6.3 Playlist-based generation of segment list 51
6.6.4 Media segment list restrictions 51
6.7 Rate adaptation 52
6.8 Seeking 53
6.9 Support for trick modes 53
6.10 Stream switching 54
6.11 Client support for dependent representations 54
6.11.1 General 54
6.11.2 Client trick-mode support using SubRepresentations 55
6.12 Events 55
6.12.1 General processing 56
6.12.2 Inband events 56
7 Extending DASH 56
7.1 Extension of MPD Schema in external namespace 56
7.1.1 General 56
7.1.2 Example 56
Bibliography 58

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This second edition cancels and replaces the first edition (ISO/IEC TR 23009-3:2015), which has been technically revised.

It also incorporates new features from ISO/IEC 23009-1:2014/Amd 1:2015 and ISO/IEC 23009-1:2014/Cor 1:2015.

ISO/IEC TR 23009 consists of the following parts, under the general title *Information technology — Dynamic adaptive streaming over HTTP (DASH)*:

— *Part 1: Media presentation description and segment formats*

— *Part 2: Conformance and reference software*

— *Part 3: Implementation guidelines*

— *Part 4: Segment encryption and authentication*

Introduction

This part of ISO/IEC TR 23009 provides guidelines for implementation and deployment of streaming media delivery systems based on the ISO/IEC 23009 series. These guidelines include the following:

— guidelines for streaming content generation;

— guidelines for implementation of streaming clients;

— guidelines for deployment of systems designed based on the ISO/IEC 23009 series.

Information technology — Dynamic adaptive streaming over HTTP (DASH) —

Part 3: Implementation guidelines

1 Scope

This part of ISO/IEC TR 23009 provides technical guidelines for implementing and deploying systems based on ISO/IEC 23009-1.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 13818-1, *Information technology — Generic coding of moving pictures and associated audio information — Part 1: Systems*

ISO/IEC 13818-2, *Information technology — Generic coding of moving pictures and associated audio information — Part 2: Video*

ISO/IEC 14496-2, *Information technology — Coding of audio-visual objects — Part 2: Visual*

ISO/IEC 14496-3, *Information technology — Coding of audio-visual objects — Part 3: Audio*

ISO/IEC 23003-1, *Information technology — MPEG audio technologies — Part 1: MPEG Surround*

ISO/IEC 23003-3, *Information technology — MPEG audio technologies — Part 3: Unified speech and audio coding*

ISO/IEC 23009-1, *Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats*

ITU-T Rec. H.264 | ISO/IEC 14496-10, *Information technology — Coding of audio-visual objects — Part 10: Advanced Video Coding*

ITU-T Rec. H.265 | ISO/IEC 23008-2, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 2: High efficiency video coding*

3 Terms, definitions, and abbreviated terms

For the purposes of this document, the terms and definitions given in ISO/IEC 23009-1; the video coding terms defined in ISO/IEC 13818-2, ISO/IEC 14496-2, ITU-T Rec. H.264 | ISO/IEC 14496-10 and ITU-T Rec. H.265 | ISO/IEC 23008-2; and the audio coding terms defined in ISO/IEC 13818-1, ISO/IEC 14496-3, ISO/IEC 23003-1 and ISO/IEC 23003-3 apply.

4 General

4.1 System overview

Figure 1 shows a typical deployment scenario for Dynamic Adaptive Streaming over HTTP (DASH). The media encoding process generates segments containing different encoded versions of one or several of the media components of the media content. Each segment contains streams required for decoding and displaying a time interval of the content. The segments are then hosted on one or several media origin servers along with a Media Presentation Description (MPD) file. The media origin server may be a plain HTTP server conforming to RFC2616. The MPD information provides instructions on the location of segments as well as the timing and relation of the segments, i.e. how they form a media presentation. Based on this information in MPD, a DASH streaming client requests the segments using HTTP GET or partial GET methods. The client fully controls the streaming session, i.e. it manages the on-time request and smooth playback of the sequence of segments, potentially adjusting bitrates or other attributes, e.g. to react to changes of the device state or the user preferences.

As long as the MPD provides RESTful HTTP-URIs for the Segment locations, the HTTP-based delivery infrastructure may be kept unaware of the actual data that is delivered. This feature permits the reuse of existing HTTP caches and Content Distribution Networks (CDNs) for massively scalable Internet media distribution.

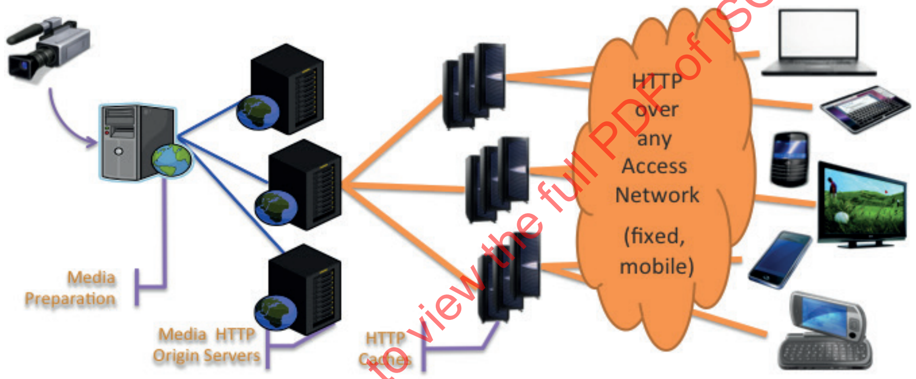

HTTP over any Access Network (fixed, mobile)
Media Preparation
Media HTTP Origin Servers
HTTP Caches

Figure 1 — Example of DASH-based media distribution architecture

4.2 Normative parts

The ISO/IEC 23009 specification serves as an enabler for Dynamic Adaptive streaming over HTTP. It does not specify a full end-to-end service, but rather base building blocks to enable efficient and high-quality streaming services over the Internet. Specifically, ISO/IEC 23009-1 defines two formats as shown in Figure 2.

— The Media Presentation Description (MPD) describes a Media Presentation, i.e. a bounded or unbounded presentation of media content. In particular, it defines formats to announce resource identifiers for Segments as HTTP-URLs and to provide the context for these identified resources within a Media Presentation.

— The Segment format specifying the format of the entity body of an HTTP response to an HTTP GET request or a partial HTTP GET, with the indicated byte range through HTTP/1.1 as defined in RFC 2616, to a resource identified in the MPD.

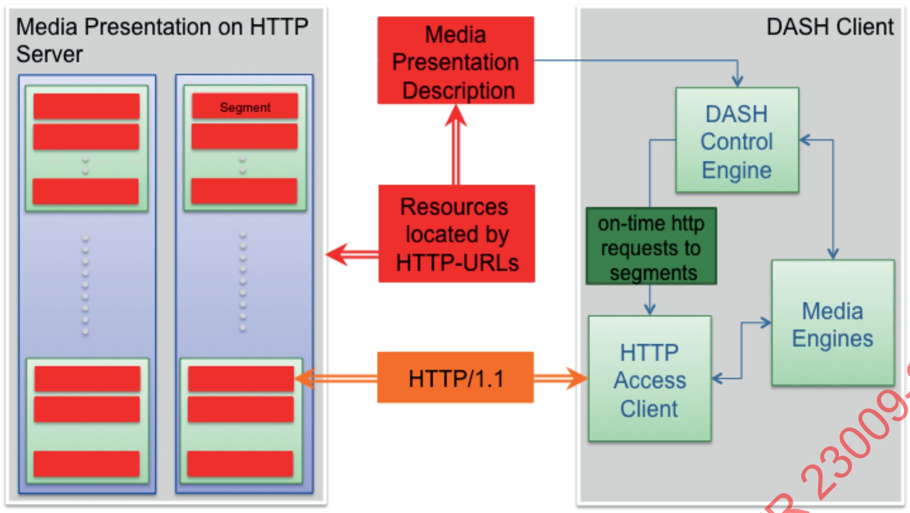

Media Presentation on HTTP Server
Segment
Media Presentation Description
Resources located by HTTP-URLs
HTTP/1.1
DASH Client
DASH Control Engine
on-time http requests to segments
Media Engines
HTTP Access Client

Figure 2 — Standardized aspects in DASH

NOTE Normative components are marked in red.

Other aspects, such as client implementations of control and media engines, are not defined as normative parts of the ISO/IEC 23009 specification.

4.3 Main design principles

4.3.1 Common timeline

ISO/IEC 23009 requires encoded versions of media content components (e.g. video, audio) to have a common timeline. The presentation time of access units within the media content is mapped to a global common presentation timeline, referred to as Media Presentation Timeline. This allows synchronization of different media components and enables seamless switching between different encoded versions of media content.

4.3.2 Data model

In ISO/IEC 23009, the organization of a multimedia presentation is based on a hierarchical data model shown in Figure 3.

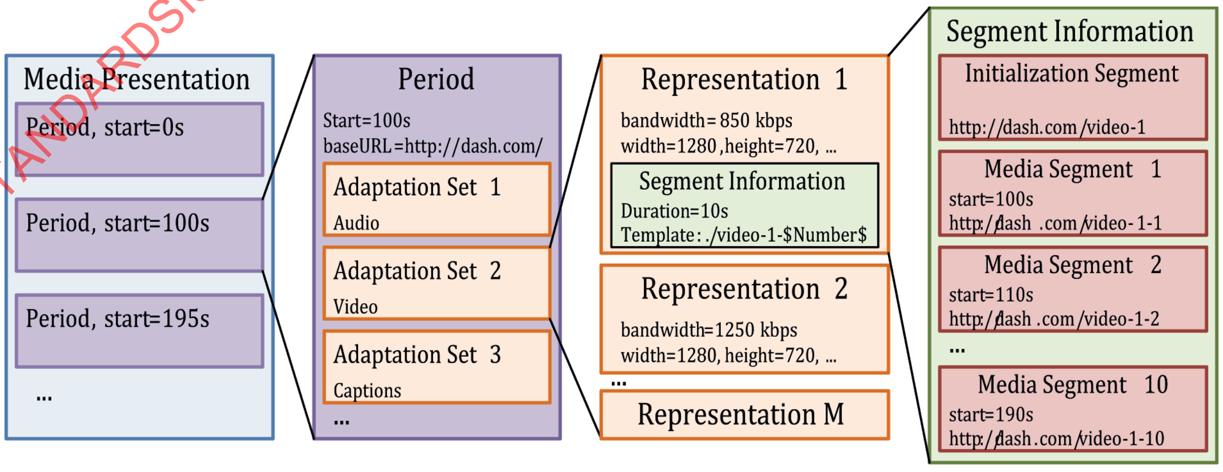


Figure 3 — DASH hierarchical data model

This model consists of the following elements.

— **Media Presentation Description** (MPD): Describes the sequence of Periods that make up a DASH Media Presentation.

— **Period**: Interval of the Media Presentation, where a contiguous sequence of all Periods constitutes the Media Presentation.

— **Adaptation Set**: Represents a set of interchangeable encoded versions of one or several media content components. For example, there may be an Adaptation Set for video, one for primary audio, one for secondary audio, and one for captions. Adaptation Sets may also be multiplexed, in which case, interchangeable versions of the multiplex may be described as a single Adaptation Set. For example, an Adaptation Set may contain both video and main audio for a Period.

— **Representation**: Describes a deliverable encoded version of one or several media content components. Any single Representation within an Adaptation Set should be sufficient to render the contained media content components. Clients may switch from Representation to Representation within an Adaptation Set in order to adapt to network conditions or other factors.

— **Segment**: Content within a Representation may be further divided in time into Segments of fixed or variable length. Each segment is referenced in the MPD by means of a URL. Thus, a Segment defines the largest data unit that can be accessed by means of a single HTTP request.

4.3.3 Segments

Segments contain encoded chunks of media components. They may also include information on how to map the media segments into the media presentation timeline for switching and synchronous presentation with other Representations.

4.3.3.1 Segment availability timeline

The Segment Availability Timeline is used to signal clients the availability time of segments at the specified HTTP URLs. These times are provided in wall-clock times. Before accessing the Segments at the specified HTTP URL, clients compare the wall-clock time to Segment availability times.

For on-demand content, the availability times of all Segments are identical. All Segments of the Media Presentation are available on the server once any Segment is available. Thus, the MPD is a static document.

For live content, the availability times of Segments depend on the position of the Segment in the Media Presentation Timeline. Segments become available with time as the content is produced. Thus, the MPD is updated periodically to reflect changes in the presentation over time. For example, Segment URLs for new segments may be added to the MPD; old segments that are no longer available may be removed from the MPD. Updating the MPD may not be necessary if Segment URLs are described using a template.

4.3.3.2 Segment duration

The duration of a segment represents the duration of the media contained in the Segment when presented at normal speed. Typically, all Segments in a Representation have the same or roughly similar duration. However, Segment duration may differ from Representation to Representation. A DASH presentation can be constructed with relative short segments (for example a few seconds) or longer Segments including a single Segment for the whole Representation.

Segments cannot be extended over time; a Segment is a complete and discrete unit that shall be made available in its entirety.

4.3.3.3 Sub-segments

Segments may be further subdivided into Sub-segments.

If a Segment is divided into Sub-segments, these are described by a Segment Index, which provides the presentation time range in the Representation and corresponding byte range in the Segment occupied by each Sub-segment. Clients may download this index in advance and then issue requests for individual Sub-segments using HTTP partial GET requests.

The Segment Index may be included in the Media Segment, typically at the beginning of the file. Segment Index information may also be provided in separate file containing Index Segments.

4.3.4 Segment types

4.3.4.1 General

ISO/IEC 23009-1 defines the following four types of segments:

— Initialization Segments;

— Media Segments;

— Index Segments;

— Bitstream Switching Segments.

4.3.4.2 Initialization segments

Initialization Segments contain initialization information for accessing the Representation and it does not contain any media data with an assigned presentation time. Conceptually, the Initialization Segment is processed by the client to initialize the media engines for enabling play-out of Media Segments of the containing Representation.

4.3.4.3 Media segments

A Media Segment contains and encapsulates media streams that are either described within this Media Segment or described by the Initialization Segment of this Representation or both. Media Segments must contain a whole number of complete Access Units and should contain at least one Stream Access Point (SAP) for each contained media stream. Other requirements applicable to Media Segments are described in ISO/IEC 23009-1:2014, 6.2.3.

4.3.4.4 Index segments

Index Segments contain information that is related to Media Segments, including timing and access information for Media Segments or Subsegments. An Index Segment may provide information for one or more Media Segments. The Index Segment may be media format specific and more details are defined for each media format that supports Index Segments.

4.3.4.5 Bitstream switching segments

A Bitstream Switching Segment contains data enabling switching to the Representation it is assigned to. It is media format specific and more details are defined for each media format that permits Bitstream Switching Segments. At most, one bitstream switching segment can be defined for each Representation.

4.3.5 Segment addressing schemes

ISO/IEC 23009 allows several alternative schemes for addressing Segments from an MPD. Specifically, a segment list for a Representation or Sub-Representation can be specified by either

— *SegmentBase* element, provided when Representation contains a single media Segment,

— *SegmentList* elements, providing a set of exact URL(s) for Media Segments, and

— *SegmentTemplate* element, providing a template form of URL(s) for Media Segments.

Details of these schemes are described in ISO/IEC 23009-1:2014, 5.3.9.

4.3.6 Stream access points

A Stream Access Point (SAP) is a position in a Representation enabling playback of a media stream to be started using only the information contained in Representation data starting from that position onwards (preceded by initializing data in the Initialization Segment, if any).

Stream access points are usually defined for each Media Segment, and used to support variety of streaming client operations, including the following:

— stream switching, e.g. for adaptation to changes in network bandwidth or other events;

— random access (seek and rewind operations);

— trick modes.

ISO/IEC 14496-12:2015, Annex I defines six possible types of SAPs. The mappings between SAP types and commonly used video prediction structures, such as Open GOP and Closed GOP structures are explained in 5.1.1.4.

4.3.7 Remote elements

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

A remote element has two attributes, `@xlink:href` and `@xlink:actuate`. `@xlink:href` contains the URL for the remote element entity, while `@xlink:actuate` specifies the resolution model. The value "`onLoad`" requires immediate resolution at MPD parse time (i.e., beginning of time or after an MPD update), while "`onRequest`" allows deferred resolution at a time when an XML parser accesses the remote element. While there is no explicit timing model for earliest time when deferred resolution can occur, ISO/IEC 23009-1 strongly suggests it should be close to the expected playout time of the corresponding period.

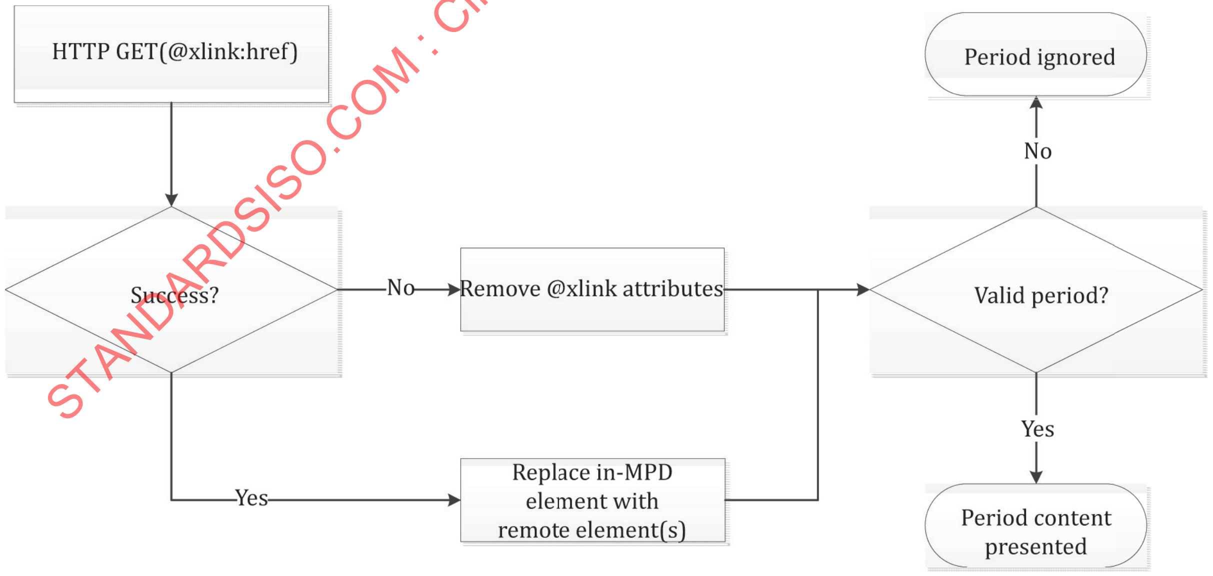


Figure 4 — XLink resolution

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

If syntactically valid remote element entity was received, it will replace the in-MPD element. "Slate" functionality, i.e. having default content that plays if dereferencing fails, can be achieved if remote periods are also valid when XLink attributes are removed.

Once a remote element entity is resolved into a fully specified element, XLink attributes that originated from the in-MPD element are removed, hence it cannot be resolved again. However, the remote element entity may contain an `@xlink:href` attribute, which contains a new XLink URL allowing repeated resolution.

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

4.3.8 Events

DASH events are messages having type, timing, and optional payload. They can appear either in MPD (as period-level event stream) or inband, as ISO-BMFF boxes of type `emsg`. The `emsg` boxes appear at the very beginning of the segment, so that DASH client will need a minimal amount of parsing to detect them.

DASH defines several events that are processed directly by a DASH client. MPD Validity Expiration, MPD Patch and MPD Update events signal to the client that the MPD needs to be updated and provide the publish time of the MPD that should be used. MPD Patch provides an additional XML patch that can be applied to the client's in-memory representation of MPD, as an alternative to requesting a new MPD. MPD Update event provides a complete new MPD.

NOTE Unless segments carrying MPD Patch or MPD Update use secure transport, such as HTTPS, a man-in-the-middle attack altering these can be mounted by a malicious entity.

User-defined events are also possible. The DASH client does not deal with them directly, they are passed to an application or discarded if there is no application willing to process these events. A possible client implementation would register callbacks to specific event types, and these would be called on arrival of these events.

In the ad insertion context, user-defined events can be used to signal information, such as cue messages.

Care should be taken to make sure that addition of events would not violate the declared bitrate constraints.

Events should be used for purposes that are related to media and are meaningful, e.g. when consumed in non-realtime applications, such as VoD, nPVR, etc.

4.3.9 General-purpose descriptors

DASH defines descriptors with clear scope. Some descriptors, such as `FramePacking`, convey specific properties of coded media that need to be supported in order to play a representation, while others (e.g. Rating) convey informational metadata. `SupplementalProperty` and `EssentialProperty` are an exception to the rule, these two general-purpose descriptors that are provided for user-defined extensions without a pre-defined scope.

`SupplementalProperty` is strictly informational, a client recognizing the specific scheme may benefit from the information provided in the descriptor, while a client that does not recognize it will function correctly.

`EssentialProperty` is used to provide information that is essential to the ability to present one or more representations. If a client does not recognize the scheme, it may be unable to present the media correctly. The `@id` property of DASH descriptors provides grouping semantics: if there are several

`EssentialProperty` descriptors with different schemes all having same `@id` value, recognizing just one of these is sufficient for correct playback.

General purpose descriptors can be present at several different levels in the MPD, from MPD down to Representation.

4.4 Background on DASH profile concept

Profiles of DASH are defined so as to enable interoperability and the signalling of the use of features. A DASH client complying with a specific profile does not necessarily have to implement features that are not required by it. By complying with a profile, a content author declares the boundaries on a set of DASH features that will be needed for successful presentation.

ISO/IEC 23009-1 defines six profiles, shown in Figure 5. Profiles are organized in two categories based on file format used for segments. Three profiles use ISO Base media file format, two profiles use MPEG-2 transport stream (TS), and full profile supports both segment formats. These profiles do not impose restrictions on codecs encapsulated by these file formats.

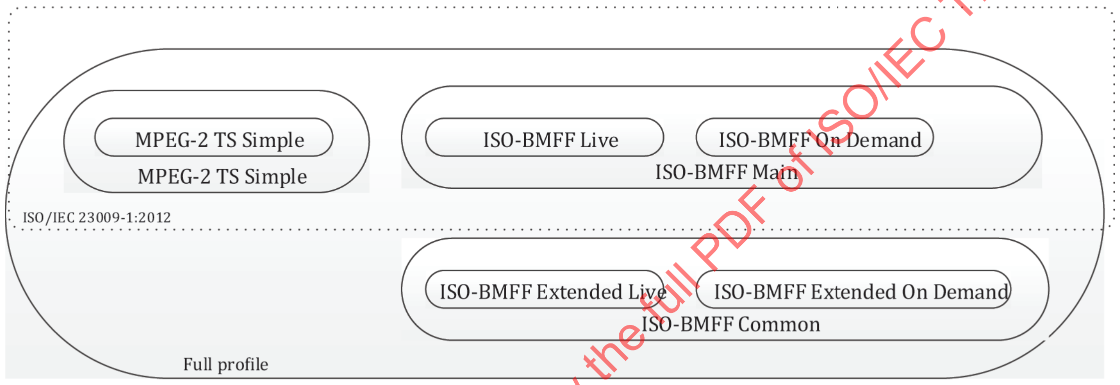


Figure 5 — DASH Profiles

In addition to six profiles defined in ISO/IEC 23009-1:2012, additional profiles may also be defined by both ISO/IEC and external entities. Such profiles may also impose constrains on codecs, resolutions, bit-rates, segment durations, and other system parameters.

Features added by the second edition of DASH, ISO/IEC 23009-1:2014, are excluded from these profiles. These are covered by the ISO-BMFF Common, Extended ISO-BMFF On Demand, and Extended ISO-BMFF Live profiles, added in ISO/IEC 23009-1:2014/Amd 1:2015.

4.5 Dynamic aspects

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

a) The content creator can announce a DASH Media Presentation for which not all content is yet available, but only gets available over time.

b) Clients are forced into some timed schedule for the playout, such that they follow the schedule as desired by the content author.

Dynamic services may be used for different types of services.

a) **Dynamic Distribution of Available Content**: Services for which content is made available as dynamic content, but the content is entirely available prior to distribution. In this case, the details of the Segments are known and can be announced in a single MPD without MPD updates.

b) **MPD-controlled Live Service**: Services for which the content is typically generated on the fly and the MPD needs to be updated occasionally to reflect changes in the service offerings. For such a service, the DASH client operates solely on information in the MPD.

c) **MPD and Segment-controlled Live**: Services for which the content is typically generated on the fly and the MPD may need to be updated on short notice to reflect changes in the service offerings. For such a service, the DASH client operates on information in the MPD and is expected to parse segments to extract relevant information for proper operation. This addresses the use cases 4 and 5, but also takes into account the advanced use cases.

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

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

— identifying the currently active Periods in the service and the Period that expresses the live edge (for more details see below);

— selecting the suitable media components by selecting one or multiple Adaptation Sets. Within each Adaptation Set selecting an appropriate Representation and identifying the live edge segment in each Representations. The client then issues requests for the segments.

The MPD may be updated on the server based on certain rules and clients consuming the service are expected to update MPDs based on certain triggers. The triggers may be provided by the MPD itself or by information included in segments. Depending on the service offering, different client operations are required as reflected in Figure 6.

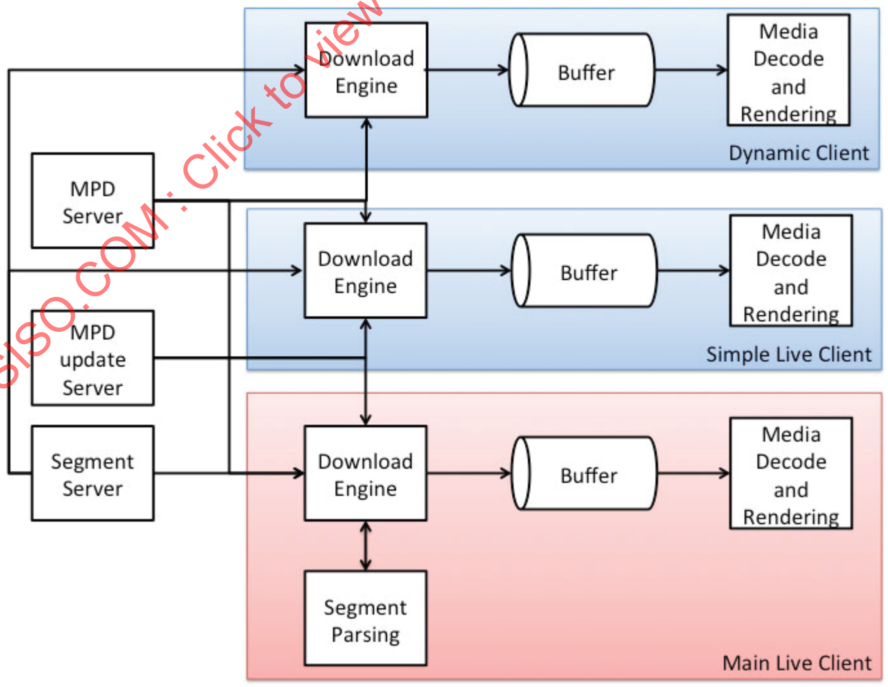


Figure 6 — Different client models

The basic operations of the different clients are as follows.

a) **Dynamic DASH Client:** Such a client is creating a list of available segments based on a single MPD. It then joins by downloading segments at the live edge or may use the available time shift buffer.

b) **Simple Live Client**: Such a client includes all features of the dynamic client. In addition, it updates the MPD based on information in the MPD in order to extend the segment list at the live edge. MPDs are refetched and revalidated when the currently available MPD expires, i.e. it can no longer be used for segment URL generation.

c) **Main Live Client**: Such a client includes all features of the dynamic client and an MPD-based DASH client. In addition, it updates the MPD based on information in the segments if the service offering provides this feature. MPDs are refetched and revalidated when the currently available MPD expires based on expiry information in the Segments.

5 Guidelines for content generation

5.1 General guidelines

5.1.1 Video content generation

5.1.1.1 General

This section presents a set of recommended practices for preparing video content for streaming using MPEG video codecs. It uses terms and concepts defined in ISO/IEC 13818-2, ITU-R Rec.H.264 | ISO/IEC 14496-10, and ITU-T Rec. H.265 | ISO/IEC 23008-2.

5.1.1.2 Enabling bandwidth adaptation

In order to support bandwidth adaptation, video content should be encoded at plurality of rates, covering the range of operational rates of the network. Such encodings should be declared as Representations with different `@bandwidth` attributes and presented in MPD as an Adaptation Set.

Video encodings within each Representation may be produced by using either Variable Bit-Rate (VBR) or Constant Bit-Rate (CBR) modes. In both cases, the `@bandwidth` attributes should be set to the maximum of the bitrate averaged over `@minBufferTime` in each Representation. When VBR encoding is used, it is also recommended to provide Index Segments, allowing client to access exact length information for each encoded segment.

The rates of different encodings should not present large gaps, as switching between them could become noticeable. Ideally, the rates in adjacent operating points should not be more than 2x apart. For example, for mobile clients connected over 3G networks and bandwidth in the range from 100 Kbps to 3 Mbps, a reasonable set of operating points may include: 100 kbps, 200 kbps, 400 kbps, 600 kbps, 1,2 Mbps, 1,8 Mbps, and 2,5 Mbps.

In order to produce encodings with different target bitrates, video encoders may be instructed to use different spatial resolutions, framerates, or other parameters needed to achieve acceptable visual quality. Such parameters should be signaled by `@width, @height, @frameRate` or other relevant attributes of Adaptation Sets and Representations. Additional properties of encoded video may be also signaled by means of EssentialProperty or SupplementalProperty descriptors with URNs and schemas defined in ISO/IEC 23001-8 or other relevant specifications.

In performing encoding, it may be desirable to minimize fluctuations of quality in the encoded content. VBR encoding is naturally more suitable for this purpose. Additionally, if system design permits the use of segments with variable durations, such durations may also be adjusted to accommodate changing complexities in video content. For example, segment boundaries may be placed at frames that are naturally encoded as I- or IDR-frames due to scene changes in video content. However, it is always a good practice to minimize deviation of segment durations to the extent possible.

5.1.1.3 Initialization segments

Initialization segments should be used to communicate sequence- and picture-level parameters needed to initialize the decoder and be able to decode media segments in the Representation.

For example, when the ITU-R Rec.H.264 | ISO/IEC 14496-10 codec is used, the initialization segment may be used to communicate SPS, and PPS elements of the bitstream.

5.1.1.4 GOP structure and Stream Access Points

In preparation of video segments, it is important to ensure that each Media Segment as at least one Stream Access Point (SAP). There are several different types of SAPs and corresponding encoding structures that can be employed.

5.1.1.4.1 SAP type 1

SAP type 1 is characterized by full alignment of time-domain parameters: $T_{EPT} = T_{DEC} = T_{SAP} = T_{PFT.}$ It can be implemented by using video structure known as “Closed GoP” (Group of Pictures). An example of such a structure is shown in Figure 7.

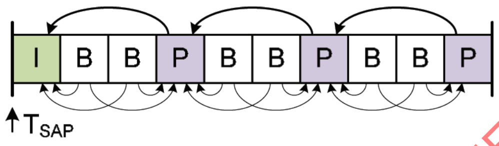


Figure 7 — Type 1 SAP implemented using Closed GOP structure

In this figure, boxes labeled as I, B, and P – correspond to I-(or IDR-), B-, and P-type pictures correspondingly. The I- (or IDR-) picture is independently decodable, while P-pictures require prior frames to be decoded first, and B-pictures require both prior and following I- or P- pictures to be decoded first.

In closed GoP structure, the I- (or IDR-) picture is transmitted first, allowing all subsequent pictures to be sequentially decoded. The first Access Unit in decoding order is also the first access unit in presentation order.

5.1.1.4.2 SAP type 2

SAP type 2 allows the presentation time of the first access unit to be delayed: $T_{EPT} = T_{DEC} = T_{SAP} < T_{PFT.}$ This can be implemented by using modified Closed GoP structure, as illustrated in Figure 8.

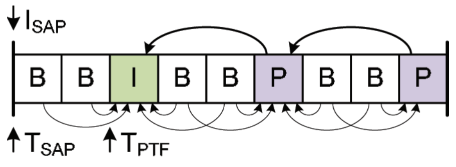


Figure 8 — Type 2 SAP implemented using modified Closed GOP structure

In this example, the first two pictures are backward predicted (syntactically they can be coded as forward-only B-pictures), and they both need 3rd picture to be decoded first. The 3rd picture is an I- (or IDR-) picture and it is transmitted in the first Access Unit.

5.1.1.4.3 SAP type 3

SAP type 3 imposes the following order on time-domain parameters: $T_{EPT} < T_{DEC} = T_{SAP} <= T_{PTF.}$ This can be implemented by using Open GoP structure, as illustrated in Figure 9.

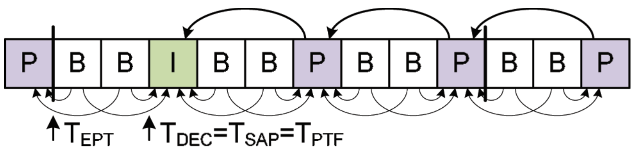


Figure 9 — Type 3 SAP implemented using Open GoP structure

In this example, Access Units corresponding to first 2 B-pictures cannot be decoded by using data within a segment, as they also rely on P-picture from a previous segment.

Open GoP structure has the advantage of being most efficient from coding efficiency standpoint. It can be used to implement random access in a single Representation. In order to enable switching between different Representations with Open GoP-based access points, such Representations should be prepared using identical codecs, resolutions, and prediction structures (codecs should have identical states of Decoded Picture Buffers).

5.1.2 Audio content generation

5.1.2.1 General

This section presents a set of recommended practices for preparing audio content for streaming when using MPEG audio codecs. It uses terms and definitions provided in ISO/IEC 13818-3, ISO/IEC 14496-3, ISO/IEC 23003-1, and ISO/IEC 23003-3.

5.1.2.2 Enabling bandwidth adaptation

In order to support bandwidth adaptation, audio content should be encoded at a plurality of rates, covering the range of operational rates of the network. Such encodings should be presented as Representations with different `@bandwidth` attributes, and presented in the MPD as an Adaptation Set.

Audio encoding characteristics such as codecs, sampling rates, and channel configurations should be communicated by means of `@codecs`, `@audioSamplingRate`, and `AudioChannelConfiguration` attributes and Elements of Representations and Adaptation Sets. Additional characteristics may be also signaled by means of EssentialProperty or SupplementalProperty descriptors with URNs and schemas defined in ISO/IEC 23001-8 or other relevant specifications. Unless declared otherwise, Media Segments carrying audio data should be understood as ones having SAP type 1, and where the first Access Unit serves as SAP.

5.1.2.3 Restrictions

To avoid discontinuities in the decoder output, the following parameters should be the same across all Representations in an Adaptation Set for a Segment:

— audio object type of the audio codec;

— channel configuration;

— sampling frequency.

Period Boundaries, however, signal a change (discontinuity) in content, e.g. ad insertion. At these boundaries, any codec reconfiguration is possible, including the above parameters.

5.1.2.4 Delay alignment

All bit streams should be delay adjusted. Encoder implementations may add additional delay depending on how they are configured (e.g. bit rate dependent Low Pass filter with varying tap count). The delay for all configurations should be pre-compensated, so that all Segments in an Adaptation Set contain the same audio information and all streams have the same framing.

5.1.2.5 AAC-LC bitrate switching

When switching between different AAC-LC Streams, the following restrictions should be taken into account, to guarantee seamless switching between different AAC-LC bit streams.

5.1.2.5.1 Window type and Window sequence

To avoid artifacts due to time aliasing components not being cancelled, window type and window shape of the AAC-LC streams should be synchronized across all bit streams at the Stream Access Point (SAP). All audio streams should use the same overlap (either long or short) and window shape (either KBD or Sine) at the Segment boundary (right window half of frame preceding SAP and left window half of SAP frame).

5.1.2.6 HE-AAC bitrate switching

As HE-AAC consists of AAC-LC and SBR audio objects, all restrictions for AAC-LC should also apply for HE-AAC. Additional adaptations for the AAC-LC core are needed to avoid small spectral holes when switching between streams with different AAC/SBR crossover frequencies. Also for the SBR part, restrictions to some SBR tools are necessary to avoid artifacts for switching between different streams.

5.1.2.6.1 Additional restrictions for AAC-LC core

To avoid gaps in the frequency range between AAC/SBR crossover frequencies of the low bit rate and high bit rate stream, the AAC-LC core bandwidth of the last frame of a segment should match the highest AAC/SBR crossover frequency of all streams in an Adaptation Set. To properly encode the additional bandwidth, extra bits are necessary. The bit reservoir control should be adapted to compensate for this extra bit demand.

5.1.2.6.2 SBR header and Time-differential coding

In contrast to the AAC configuration that is completely described by the audio specific configuration (ASC), the SBR decoder needs additional configuration parameters. These parameters are transmitted inside the SBR Header which may not be contained in every access unit (AU). The MPEG-4 standard recommends a transmission interval of 500 ms or whenever an instantaneous change of header parameters is required (see ISO/IEC IS 14496-3:2009, 4.5.2.8.2.1).

To allow seamless switching of HE-AAC bit streams, it is necessary to transmit an SBR header in the first Access Unit of a Segment (SAP frame). As the MPEG-4 Audio Conformance (ISO/IEC IS 14496-26) forbids the use of tools that rely on preceding frames for frames containing an SBR Header, with this restriction it is also assured that the SAP frame can be completely decoded and processed.

5.1.2.6.3 SBR frame class

SBR envelopes can reach over frame borders, i.e. “VARVAR” and “FIXVAR” frames may overlap the SBR frame border. A SAP Frame should always start with a FIX border (“FIXVAR” or “FIXFIX”) to make sure all necessary information to fully decode the audio contained in that frame is available. Consequently, the last frame in a segment (the frame before the SAP) should end with a “FIX” border (i.e. a “FIXFIX” or “VARFIX” frame).

5.1.2.7 HE-AACv2 bitrate switching

As HE-AACv2 relies on an HE-AAC core and adds the Parametric Stereo (PS) audio object, all requirements listed for AAC-LC and HE-AAC streams should also apply for HE-AACv2.

5.1.2.7.1 PS header and Time-differential coding

As with the SBR payload, also within the PS payload, Configuration Parameters may be transmitted not with every frame, but on a less regular basis. Also, time differential coding of certain parameters can be used to increase compression efficiency.

To allow for a seamless switching of HE-AACv2 bit streams, it is necessary to transmit a PS header with each SAP frame. For frames containing a PS Header, the MPEG-4 Audio Conformance forbids the use of tools that rely on past information, i.e. time differential coding of parameters.

With the above restriction, it is assured that the SAP frame can be completely decoded and processed. As HE-AACv2 Conformance requires a PS Header with every SBR header, this requirement is also implicitly inherited from the HE-AAC requirements.

5.1.2.7.2 PS tools and parameters

All PS Tools are designed to allow for continuous remapping of different configurations (e.g. frequency resolution of parameter bands). For the baseline version of the PS Tool, no extra care has to be taken at Stream Access Points.

5.1.2.8 AAC-LC/HE-AAC plus MPEG Surround bitrate switching

As MPEG Surround is based on an AAC-LC or HE-AAC core coder, all restrictions for AAC-LC and HE-AAC should also apply for MPEG Surround. Further restrictions are as follows, for details see ISO/IEC 23003-1.

The MPEG Surround data shall be conveyed in the AAC extension payload using implicit signaling. Each SAP frame shall contain the syntactic element *SpatialSpecificConfig* that contains the MPEG Surround configuration data. Additionally, the coding of SAP frames should be independent of previous frames. Therefore, the bitstream payload element *bsIndependencyFlag* should be set to 1.

The MPEG Surround tool *residual coding* employs a representation of differential signals using the AAC-LC syntax. As described in 5.1.2.5.1, the window type of the residual signal should be synchronized at the SAP.

5.1.3 Content preparation for live streaming

The DASH media presentation description and encoded segments should be prepared in conformance to timing model described in 6.4.

5.1.4 Guidelines for generation of segment file names

5.1.4.1 General

In preparation of DASH content for distribution, it is important to ensure that encoded segments are given unique names, allowing their storage in same location and referencing from an MPD file.

This section provides recommended process for generation of file names, based on mapping of content parameters to file names. This process includes mapping of static content parameters, as well as dynamic parameters, such as $number$ and $time$ enabled by `SegmentTemplate` elements.

5.1.4.2 Segment URL generation

Mapping of content parameters to URLs is accomplished by using a set of key/value pairs provided in Table 1. Such key/value pairs shall only be used right of the rightmost character “/”. The first character after “/” does not represent a key. The first key shall only be present right of the first ‘_’ which is right of the rightmost character “/”. The ordering of the key/value pairs in the string is arbitrary. The key/value pairs shall only use delimiters “__”. No delimiter is added between the key and the value. Note that the syntax and semantic is compatible with ISO/IEC 23009-1 and only provides an additional restriction to the exact syntax of the HTTP URL.

Each field starts from a 1-character prefix followed by a value derived from the appropriate MPD value. Fields are separated by the character '_', thus parsing should be done looking for the character '_', with the following character providing information on the field.

The only '.' character after the "/" character should be after the segment number/time and before the file extension. In case of ISO-BMFF, the end of a segment name shall be ".mp4" if the file contains more than one content component (e.g. multiplexed audio and video) or non-audiovisual component(s). Unmultiplexed ISO-BMFF video segments shall have the ".m4v" extension, while unmultiplexed audio shall have the extension ".m4a".

All fields should appear between the first '_' character and the first "." character. The query string may not be used for carrying fields.

In summary, the generic structure of a segment name is:

[Name]_[field0]_[field1]_[fieldN]_[v|a|...]_[number|time].mp4

Segment name should not start with the '_' character. If there is no meaningful name to be given, the string "seg" should appear at the beginning of the segment name.

The prefixes that can be used in segment names, and the corresponding MPD variables are listed in Table 1.

Table 1 — Key/value pairs for segment URL generation

Field prefix	Corresponding MPD variable	Restrictions (in 5.1.4.2.1)
M	`MPD@id`	a)
I	`Period@id`	a)
A	`AdaptationSet@id`	a)
g	`AdaptationSet@group`	
l	`AdaptationSet@lang`	c)
P	`AdaptationSet@par`	b)
R	`Representation@id`	a)
B	`Representation@bandwidth`	
W	`Common@width`	
H	`Common@height`	
s	`Common@sar`	b)
F	`Common@framerate`	b)
F	`Common@audioSamplingRate`	b)
N	`$Number$`	d),e),i)
T	`$Time$`	e)
C	`Common@codecs`	b)

5.1.4.2.1 Restrictions and processing rules

The following restrictions and processing rules should apply.

a) **`MPD`**`@id`, **`Period`**`@id`, and **`Representation`**`@id` should only contain alphanumeric characters. They should not contain a character '_'. However, when **`Representation`**`@id` is used to convey these fields, character '_' will appear as a field separator.

b) Characters '/' , ':', and '.',e.g. in **`Common`**`@framerate`, **`Common`**`@sar`, **`Common`**`@codecs`, should be replaced with '-'. Thus, framerate of 24000/1001 should be written as _f24000-1001.

c) Whitespace-separated lists should be written as two separate fields, e.g. for lang="en es" the filename would include "_len_les".

d) Segment Number should be zero-padded to width of at least 5 digits, i.e. it should appear, for example, as `$Number%05d$`. The reasoning behind this restriction is to make sure lexical ordering and numerical ordering are identical for the whole Representation.

e) Segment number or time is the rightmost character before the file extension, i.e. a URL is generated in a way that if file extension and either `$Number$` or `$Time$` are present, only the dot-separated file extension may follow them.

f) When either `$Number$` or `$Time$` is used, the appropriate prefix ('n' or 't') Segment name contains the first character of the `@contentType` at the right-most position left of the segment number separated from it with an '_', e.g. `_v_$Number$`.

g) **`Common`**`@bandwidth` should be expressed in kilobits per second, and end with the character "k".

h) Naming is case-insensitive.

i) The Initialization Segment contains `0` at the `$Number$` position, and `$Number$` is zero-padded.

5.1.4.3 Examples

5.1.4.4 Segment file names

www.example.com/SomeMovie_w720_h480_b500k_V_n00278.m4v

www.example.com/SomeMovie_f44100_b32k_A_n00172.m4a

www.example.com/SomeMovie_f24_w1920_h1080_cavc1-648028_b4000k_V.m4v

5.1.4.5 MPD syntax

Use **`BaseUrl`** and **`Representation`**`@id` for inserting most of the time-independent fields above. This will make MPD more readable.

Note that including all fields described in 5.1.4.2 may make segment names less readable. It is recommended to include all fields that differ in value among the representations. It is also recommended to omit variables that don't differ between representations and do not contribute to readability.

An MPD segment shown in Figure 10 below illustrates this approach.

```
<BaseURL>SomeMovie_</BaseURL>

<AdaptationSet
  mimeType="video/mp4"
  codecs="avc1.648028"
  frameRate="24"
  segmentAlignment="true"
  bitstreamSwitching="true"
  startWithSAP="2"
  subsegmentStartsWithSAP="2">

  <SegmentTemplate
    media="$RepresentationID$_V_n$Number%05$.m4v"
    initialization="$RepresentationID$_V_init.m4v"
    duration="4"
    startNumber="1"/>

  <Representation
    id="b4000K_w1920_h1080_f24_cavc1-648028"
    bandwidth="4000000" width="1920" height="1080" />
```

Figure 10 — Example of MPD using segments with generic file names

5.2 Guidelines for ISO-BMFF content generation

5.2.1 On-demand streaming

5.2.1.1 Video on demand distribution

5.2.1.1.1 Use case

On-demand streaming of multimedia content encoded and encapsulated in ISO Base media file formats. The multimedia content may include video and may be accompanied by audio and subtitle tracks in several languages. Content protection schemes may also be applied.

The encodings of both video and audio tracks can be done at multiple bitrates and different resolutions (sampling rates) to serve users accessing content from different devices and network environments.

5.2.1.1.2 MPD authoring

The MPD file for this use case should be prepared in accordance with constraints for ISO Base media file format On Demand profile, as specified in ISO/IEC 23009-1:2014, 8.1, 8.3.1, and 8.3.2.

When multiple language tracks are provided they should be included in different Adaptation Sets, with *lang* parameters set to specify each language.

An example MPD file illustrating such use scenario is shown in Figure 11. This MPD document describes content available from two sources (cdn1 and cdn2) that has audio available in English or French at rates of 64 kbits and 32 kbits and subtitles in German. Six versions of the video are provided at bitrates between 256 kbit/s and 2 Mbit/s in different spatial resolutions. Content protection is applied.

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:mpeg:DASH:schema:MPD:2011"
  xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011"
  type="static"
  mediaPresentationDuration="PT3256S"
  minBufferTime="PT1.2S"
  profiles="urn:mpeg:dash:profile:isoff-on-demand:2011">

  <BaseURL>http://cdn1.example.com/</BaseURL>
  <BaseURL>http://cdn2.example.com/</BaseURL>

  <Period>
    <!-- English Audio -->
    <AdaptationSet mimeType="audio/mp4" codecs="mp4a.0x40" lang="en" SubsegmentAlignment="true">
      <ContentProtection schemeIdUri="urn:uuid:706D6953-656C-5244-4D48-656164657221"/>
      <Representation id="1" bandwidth="64000">
        <BaseURL>7657412348.mp4</BaseURL>
      </Representation>
      <Representation id="2" bandwidth="32000">
        <BaseURL>3463646346.mp4</BaseURL>
      </Representation>
    </AdaptationSet>
    <!-- French Audio -->
    <AdaptationSet mimeType="audio/mp4" codecs="mp4a.40.2" lang="fr" SubsegmentAlignment="true">
      <ContentProtection schemeIdUri="urn:uuid:706D6953-656C-5244-4D48-656164657221"/>
      <Role schemeIdUri="urn:mpeg:dash:role" value="dub"/>
      <Representation id="3" bandwidth="64000">
        <BaseURL>3463275477.mp4</BaseURL>
      </Representation>
      <Representation id="4" bandwidth="32000">
        <BaseURL>5685763463.mp4</BaseURL>
      </Representation>
    </AdaptationSet>
    <!-- Timed text -->
    <AdaptationSet mimeType="text/mp4" codecs="3gp.text" lang="fr" lang="de">
      <Role schemeIdUri="urn:mpeg:dash:role" value="subtitle"/>
      <Representation id="5" bandwidth="256">
        <BaseURL>796735657.mp4</BaseURL>
      </Representation>
    </AdaptationSet>
    <!-- Video -->
    <AdaptationSet mimeType="video/mp4" codecs="avc1.4d0228" SubsegmentAlignment="true">
      <ContentProtection schemeIdUri="urn:uuid:706D6953-656C-5244-4D48-656164657221"/>
      <Representation id="6" bandwidth="256000" width="320" height="240">
        <BaseURL>8563456473.mp4</BaseURL>
      </Representation>
      <Representation id="7" bandwidth="512000" width="320" height="240">
        <BaseURL>56363634.mp4</BaseURL>
      </Representation>
      <Representation id="8" bandwidth="1024000" width="640" height="480">
        <BaseURL>562465736.mp4</BaseURL>
      </Representation>
      <Representation id="9" bandwidth="1384000" width="640" height="480">
        <BaseURL>41325645.mp4</BaseURL>
      </Representation>
      <Representation id="A" bandwidth="1536000" width="1280" height="720">
        <BaseURL>89045625.mp4</BaseURL>
      </Representation>
      <Representation id="B" bandwidth="2048000" width="1280" height="720">
        <BaseURL>23536745734.mp4</BaseURL>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Figure 11 — Example MPD file for on-demand video streaming using ISOMBFF

5.2.1.1.3 Segment generation

Media segments for this use case should be prepared in accordance with constraints for ISO Base media file format On Demand profile, specified in ISO/IEC 23009-1:2014, 8.1, 8.3.1, and 8.3.3.

In order to enable bandwidth adaptation, video and audio segments can be encoded at multiple bitrates, representative of typical connection speeds and device capabilities of intended audience.

Such encodings should be declared as Representations. Sets of Representations for which switching is enabled should be declared as Adaptation Sets.

In the simplest case, all Segments can be encoded as Self-Initializing Segments. More generally, encoding may also include Initializing Segments and/or Stream-Switching Segments, as described in ISO/IEC 23009-1:2014, 5.3.9.5.2 and 5.3.9.5.5, correspondingly.

Figure 12 shows an example of the structure of encoded Segments for an Adaptation Set containing three Representations. As shown in the figure, the Segments are Self-Initializing Segments and the Subsegments are aligned. Subsegments begin with Stream Access Points. Each of the 'moov' box in the Self-Initializing Segments contain sample entries and coding information for the data following in the Subsegments of the corresponding representation.

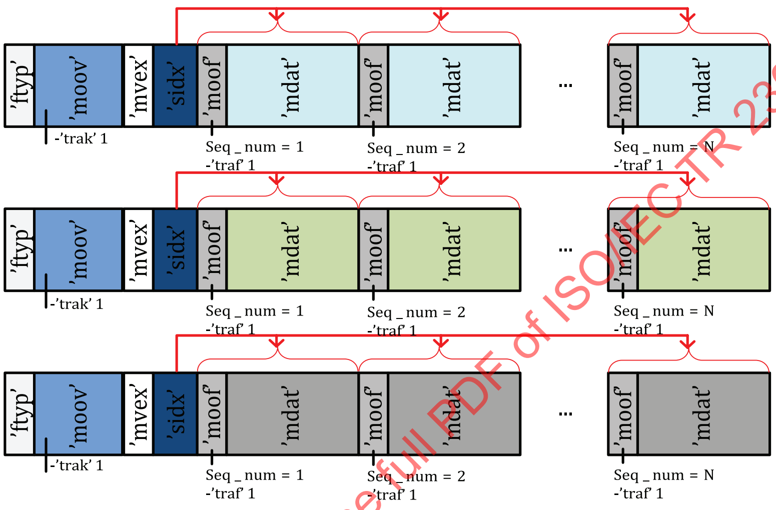


Figure 12 — Example set of Segments for on-demand video streaming using ISOMBFF

5.2.1.2 Video on demand distribution using dependent representations

5.2.1.2.1 Use case

Video on demand distribution targeting clients with capability to decode and playback content encoded using scalable video codecs, such as MPEG-4 SVC.

Content generation for this use case must provide support for On-Demand services, permitting scalable and efficient use of HTTP servers and simplifying seamless switching. This can be accomplished by defining Representation that consists of a single Self-Initializing Indexed Segment and Subsegments that are aligned across Representations within an Adaptation Set and start with Stream Access Points of type 1, 2, or 3.

5.2.1.2.2 MPD authoring

The MPD file for this use case should be prepared in accordance to general constraints for ISO Base media file format On Demand profile, specified in ISO/IEC 23009-1:2014, 8.1, 8.3.1, and 8.3.2.

The dependent Representations should include the `@dependencyId` attribute. Additionally, the `@bandwidth` attribute of dependent Representations should refer to the whole Representation result of combining the dependent Representations with their complementary Representations, i.e. the cumulative bandwidth.

5.2.1.2.3 Segment generation

Media segments for this use case should be prepared in accordance to general constraints for ISO Base media file format On Demand profile, specified in ISO/IEC 23009-1:2014, 8.1, 8.3.1, and 8.3.3.

In addition, segments encoded as dependent representations should fulfil the following constraints.

a) The Segments in the dependent Representations are simple Indexed Media Segments as defined in ISO/IEC 23009-1:2014, 6.3.4.3. They do not require the boxes contained in the Initialization segment (e.g. 'moov', 'mvex', etc.) since they must be combined at the DASH Client with their complementary Representations as detailed in ISO/IEC 23009-1:2014, 5.3.5.3, where the initialization information is contained. Therefore, they do not contain a 'ftyp' box but a 'styp' box, where the 'msix' brand is contained as a compatible brand.

b) The Segment in the base Representations, i.e. the non-dependent Representations without `@dependencyId` attribute, is a Self-Initializing Indexed Segment, i.e. it conforms to the concatenation of Initialization Segment and Index Media Segments without the 'styp' box as defined in ISO/IEC 23009-1:2014, 6.3.5.2. It contains all the 'trak' boxes with the corresponding sample descriptions for all the dependent Representations that depend on this base Representation.

c) The 'trak' box corresponding to a media component also contained in a complementary Representation, if any, shall contain the 'tref' box indicating the track IDs on which the track in the dependent Representation depends on, and indicating also the type of dependency, as for instance 'scal' for SVC.

d) The sequence_number in the 'mfhd' of the movie fragments in the Subsegments fulfils:

$$Lmf\left(S_i, R_n\right) < Fmf\left(S_i, R_j\right) \qquad \forall i \cap \forall n \mid 0 \le n < j \le N \tag{1}$$

assuming that the dependent Representation has `@id` equal to R_N and `@dependencyId` equal to "R_0 R_1 R_2 R_3 ... R_{N-1}," and where $Lmf(S_i,R_n)$ is the sequence_number in the 'mfhd' of the last movie fragment in Subsegment *i* in the Representation with `@id` equal to R_n and $Fmf(S_i,R_j)$ is the sequence_number of in the 'mfhd' of the first movie fragment in Subsegment *i* in the Representation with `@id` equal to R_j.

Figure 13 shows an example of an Adaptation Set containing dependent Representations. This example includes a video stream with 3 layers and an audio stream. The presentation is split into three Representations. The Representation at the bottom contains the base layer of the video and the audio stream. In the 'moov' box, all the tracks are described: the base layer track and the audio track contained within the Representation and also the tracks for the higher layers (e.g. enhancement layers of SVC or enhancement views of MVC) of the video in the dependent Representations are described.

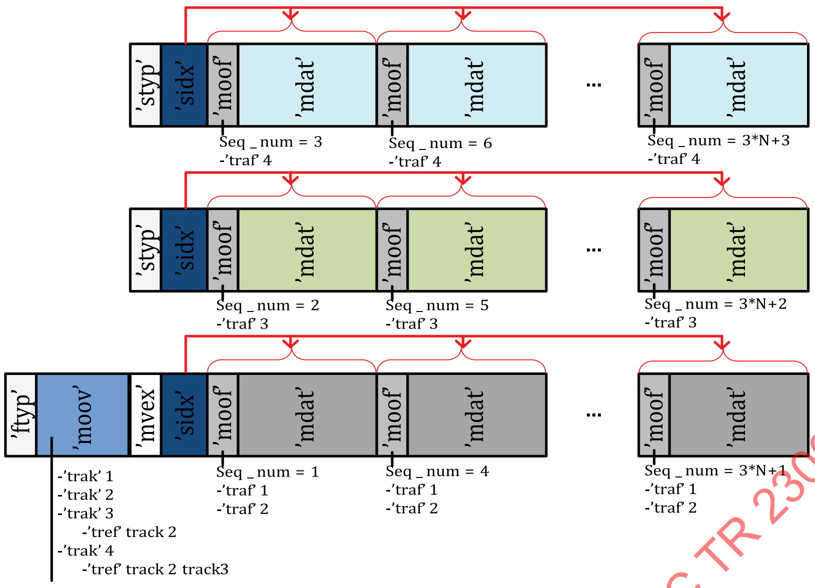


Figure 13 — Segments for an Adaptation Set containing dependent Representations

As shown in Figure 13, the 'trak' boxes corresponding to the tracks containing higher layers, include the 'tref' box showing the dependencies of these tracks to lower tracks. Furthermore, the sequence_numbers in the 'mfhd' of the 'moof' boxes follow the constraints expressed in Formula (1).

5.2.2 Live streaming

5.2.2.1 Live video distribution

5.2.2.1.1 Use cases

A service provider wants to provide a live soccer event using DASH that can potentially be accessed by millions of users. The service provider provides redundant infrastructure in terms of encoders and servers to enable a seamless switch-over in case any of the components fail during the live event or get overloaded.

Anna accesses the service in the bus with her mobile DASH-enabled device and the service is available immediately.

Continuing the use case, across from her sits Paul, who watches the event on his DASH-enabled laptop. A goal is scored and both, despite watching on different screens, celebrate this event at the same time.

Continuing the use case, other people that follow the game on a 3GPP Rel-6 PSS terminal observe the goal within a similar time.

Continuing the use case, another goal is scored. Paul tells Anna that the first goal in the game was even more exciting and Anna uses the offering that she can view the event 30 minutes back in time on her DASH-enabled device. After having seen the goal, she goes back to the live event.

Continuing the use case, the football match gets into overtime, the star player of CF Anolecrab, Lenoil Issem, is brought into the game by the coach of the year, Aloidraug, hits twice the post, but cannot score. Due to the extraordinary tension in the match, more and more users join such that the service provider requires migrating the service to the redundant infrastructure without interrupting the service to the users.

Continuing the use case, finally penalty shooting is necessary. The live event is interrupted by a short break during which advertisement is added. The exact timing of the ad breaks is unknown due to the extra time of the extension and the start of the penalty shooting is delayed.

5.2.2.1.2 MPD generation

MPDs files for this use case should be prepared in accordance to general constraints for ISO Base media file format Live profile, specified in ISO/IEC 23009-1:2014, 8.1, 8.4.1, and 8.4.2.

6.4 describes time model that should be taken into account in producing MPD files.

5.2.2.1.3 Segment generation

Media segments for this use case should be prepared in accordance to general constraints for ISO Base media file format Live profile, specified in ISO/IEC 23009-1:2014, 8.1, 8.4.1, and 8.4.3.

Segment generation process should be arranged such that it is possible to implement timely MPD updates and follow time model described in 6.4.

5.2.2.2 Live video distribution using dependent representations

5.2.2.2.1 Use case

Live video distribution targeting clients with capability to decode and playback content encoded using scalable video codecs, such as MPEG-4 SVC.

Content generation for this use case should provide support for Live services and enable seamless switching. This can be accomplished by defining Representation consisting of several Media Segments of relatively short duration for achieving low latency, which are aligned across Representations within an Adaptation Set and start with Stream Access Points of types 1, 2, or 3.

5.2.2.2.2 MPD generation

The MPD file for this use case should be prepared in accordance to general constraints for ISO Base media file format Live profile, specified in ISO/IEC 23009-1:2014, 8.1, 8.4.1, and 8.4.2.

In addition, the dependent Representations shall include the `@dependencyId` attribute and the `@bandwidth` attribute shall refer to the whole Representation result of combining the dependent Representations with their complementary Representations, i.e. the cumulative `@bandwidth`.

Furthermore, the Initialisation Segment addressed in the MPD shall be included at least in the base Representation, i.e. Representation without `@dependencyId`. If included also for the dependent Representation, the pointed URL should be the same as the one for the base Representation, i.e. the Initialisation Segment is the same for all dependent Representation and their complementary Representations.

5.2.2.2.3 Segment generation

Media segments for this use case should be prepared in accordance to general constraints for ISO Base media file format Live profile, specified in ISO/IEC 23009-1, 8.1, 8.4.1, and 8.4.3.

In addition, segments encoded as dependent representations shall fulfil the following constraints.

a) The ‘track’ box corresponding to a media component also contained in a complementary Representation, if any, shall contain the ‘tref’ box indicating the track on which the track in the dependent Representation depends on, and indicating also the type of dependency, as for instance ‘scal’ for SVC.

b) The sequence_number in the ‘mfhd’ of the movie fragments in the segments fulfil Formula (1). Note that in absence of Segment Indexes, i.e. ‘sidx’ boxes, the segment itself is considered as a single Subsegment.

5.2.3 Enabling trick modes

5.2.3.1 Use case

Lisa consumes the latest series of the show “Lost” in SD coded at a bitrate of 2 MBit/s that is distributed to a DASH-ready Client set. Her client is equipped with a H.264/AVC video decoder that is capable to handle H.264/AVC High Profile level 3.0. All of a sudden, the phone rings and she pauses the service.

After the phone call, she resumes the service, but realizes that she wants to go backward in time, as she cannot remember the start of the scene. She seeks backward to the last scene changes and resumes the service from there.

After a while, she needs to leave for her football practice and she decides to continue to watch the movie from her smart phone with H.264/AVC CBP level 1,3. She enters the service and does a fast-forward 64-times of the original speed to the position where he stopped on the TV set. Once she is close, she reduces the search speed gradually down until she recognizes the position. Once the position found, she resumes the service at normal playback speed.

She meets her friend Max and pauses the service. She remembers the great scene in the show wants to share the scene with her friend. She seeks backward in-time and finally gets to the scene and shares it with her friend.

5.2.3.2 MPD authoring

The MPD file for this use case should be prepared in accordance to general constraints for ISO Base media file format On Demand profile, specified in ISO/IEC 23009-1:2014, 8.1, 8.3.1, and 8.3.2.

In addition, the following conditions should be satisfied:

— the SubRepresentation element should be contained at the Representation;

— the **SubRepresentation**@level should be present;

— the **SubRepresentation**@dependencyLevel should be provided to indicate the dependencies among SubRepresentations.

5.2.3.3 Segment generation

Media segments for this use case should be prepared in accordance to general constraints for ISO Base media file format On Demand profile, specified in ISO/IEC 23009-1:2014, 8.1, 8.3.1, and 8.3.3.

In addition, the following conditions should be satisfied.

a) The Initialization Segment should contain the Level Assignment (‘leva’) box with the same levels as provided in **SubRepresentation**@level.

b) All Media Segments should conform to Sub-Indexed Media Segments as defined in ISO/IEC 23009-1:2014, 6.3.4.4 and therefore should include ‘sims’ as compatible brand in the ‘styp’ box.

c) If the SubRepresentations defined by the levels in the ‘leva’ box have assignment type equal to 0 or 1 for a track, the Media Segments should contain the ‘sbgp’ (sample to group) box in the corresponding ‘traf’ and the ‘sgpd’, in case the corresponding ‘sgpd’ is not included in the ‘stbl’ in the Initialization Segment.

d) If the SubRepresentations defined by the levels in the ‘leva’ box have assignment type equal to 2, a single movie fragment is contained in the Subsegment and each of the level contains data of a single track of the tracks indicated in the ‘trak’ boxes in the ‘moov’ box.

e) If the SubRepresentations defined by the levels in the ‘leva’ box have assignment type equal to 3, more than one movie fragment is contained in the Subsegment and each of the level contains data of a movie fragment.

f) If the SubRepresentations defined by the levels in the 'leva' box have assignment type equal to 4 for a track, the Media Segments shall contain the 'sbgp' (sample to group) box in the corresponding 'traf' and the 'sgpd', in case the corresponding 'sgpd' is not included in the 'stbl' in the Initialization Segment. Furthermore, the Media Segment shall contain a 'udta' box with a 'strk' box. The 'stsg' box in the 'strd' of the 'strk' contains information to identify the sample grouping information in the 'sbgp' box.

g) Data from lower levels should not depend on data in higher levels.

There are four possibilities of generating the segments in order to allow for trick modes, i.e. c), d), e), and f).

When c) is considered and assuming the trick mode is performed only for the video media component, there is a single track with sample groups for describing the different level (e.g,. the 'tele' sample group). In this case, as well as if level definition is based on subtracks f), it is necessary to arrange all the samples belonging to each of the leves at the beginning of the Subsegment. As an example, Figure 14 shows how this can be done for fast forwarding using the sample grouping 'tele' for a video stream encoded with AVC with GOP size 4 using bi-predictive hierarchical pictures, i.e. with Structure $IB_1B_0B_1P$... in presentation order.

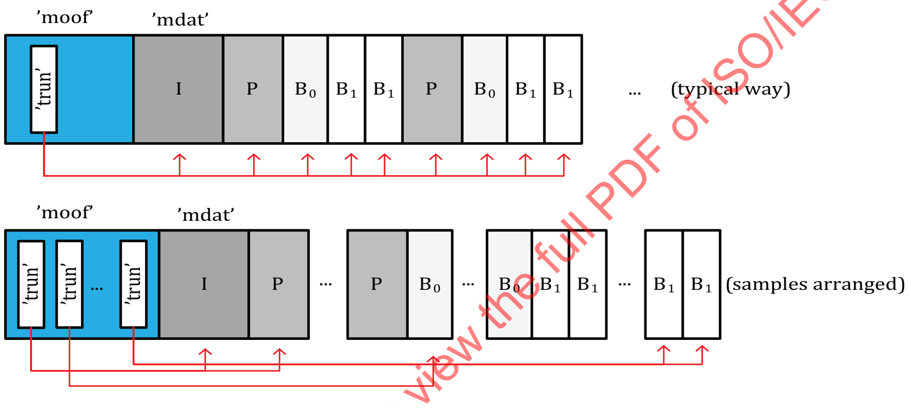


Figure 14 — Movie fragment format for arranged samples for easing fast forward with 'ssix' box

Since it is necessary to group the samples in temporal order, it is necessary to split the 'trun' in multiple 'trun'-s. For such an arrangement of the samples in an order different from the decoding order, it is necessary to add multiple 'trun' boxes in order to still provide the correct decoding time. Whenever two contiguous samples in the 'mdat' do not have decoding time following each other, a new 'trun' is needed. Then the different levels could be described e.g. as level 0 containing I and P frames, level 1 containing B_0 frames, level 2 B_1 frames, and so forth.

If d) and e) are considered, i.e. 'leva' box with assignment type 2 or 3, the usage of extractors would be needed for preparing the content for allowing fast forward trick mode, as explained in 6.9 for a more general purpose.

5.2.4 Support for SubRepresentations

5.2.4.1 Use case

Sub-representations can be used with any profile defined in ISO/IEC 23009-1. In order to access the sub-representations, it is necessary to download the 'ssix' boxes which determine the byte-ranges at which the partial data of a Subsegment that corresponds to a given subrepresentation can be accessed. Therefore, sub-representations are more suitable for VoD or Live with type 'static'. Trick modes (described in 6.9) are a particular use case of sub-representations for a specific purpose.

5.2.4.2 MPD authoring

Same rules as described in 5.2.3.2 apply.

5.2.4.3 Segment generation

Same rules as described in 5.2.3.3 apply.

In Figure 15, an example of the format segment for supporting SubRepresentations for an assignment type other than 3 is shown. In this case, the 'moof' box contains all the tracks and a Subsegment should consist of a single movie fragment. The yellow arrows and the dashed lines correspond to the position until which the data belonging to the first level is present, which is indicated in the 'ssix'. In this example, only two levels are considered and the second expands until the end of the Subsegment (in this case, movie fragment).

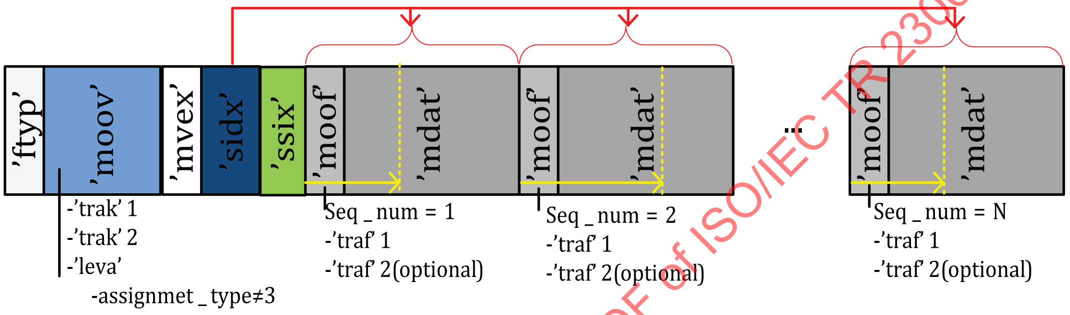


Figure 15 — Example of usage of 'ssix' box for Sub-Representations for self-initialising segment with assignment type in 'leva' box other than 3

In Figure 16, an example of a Segment format for assignment type equal to three is shown. As it can be seen in this figure, each of the movie fragments contained within a Subsegment contains data from different tracks. In this case, the byte ranges provided by the 'ssix' box should contain whole numbers of movie fragments.

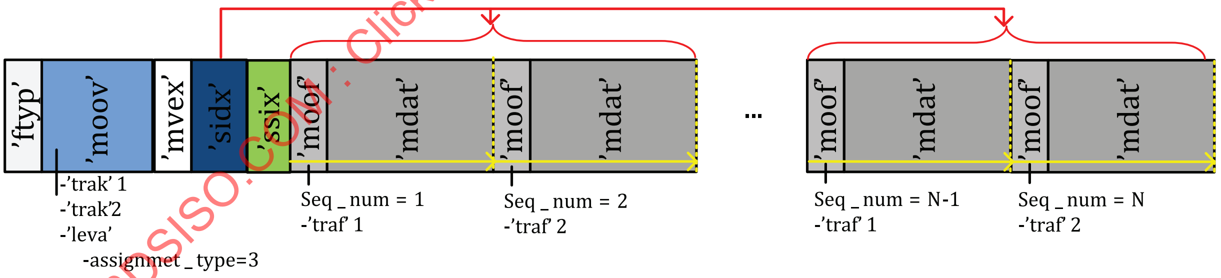


Figure 16 — Example of usage of 'ssix' box for Sub-Representations for self-initialising segment with assignments type in 'leva' box equal to 3

In general, when the tracks are used to perform sub-representation extraction (e.g. for trick modes), if several tracks describe one media component, extractors are used and only one track of those is played accessing to the samples in other tracks by reference by the extractors. The usage of extractors should be done very carefully if combined with sub-representations. If extractors are used in higher levels pointing to lower levels, there would not be any problem at the client side but if extractors are used in the segments, and these are stored in the lower level pointing to data in higher levels, DASH Clients may try to access non existing data. Therefore, special care should be taken to use extractors in lower levels if assignment type other than 3 is used and the padding_flag in the 'leva' box is set.

5.2.5 Enabling delivery format to storage file format conversion

5.2.5.1 Use case

During a DASH session, there is often a need to convert received media segments from delivery format to storage file format so that they can form a media file that can be stored on a storage device in such a way that it is syntactically valid and can be rendered by a media player that does not have DASH client functionality.

This function can be achieved by using the `@bitstreamSwitching` attribute in the MPD document. Media segments received for periods where this attribute is set to TRUE can be simply concatenated to form a bitstream that conforms to the media formats in use.

This section provides guidelines for implementing this function with this approach for the case of ISOBMFF based DASH services. It also provides examples to illustrate the guidelines.

5.2.5.2 MPD authoring

The `@bitstreamSwitching` attribute in the MPD for a DASH session can be set to TRUE to generate media segments that are required to support delivery to storage format conversion functionality by simple concatenation of media segments received by the client for the session.

5.2.5.3 Segment generation

If the `@bitstreamSwitching` attribute is set to TRUE, segments should meet the following conditions.

a) All the Media Segments shall meet the conditions implied when `@segmentAlignment` attribute is set to 'true' (refer to ISO/IEC 23009-1:2014, 7.3.3.2).

b) All the Representations within an Adaptation Set shall have an identical Initialization Segment.

c) The Initialization Segment shall include all the sample descriptions required to decode all the Representations within the Adaptation Set. This means that if a media content component is represented differently across all the Representations, the 'moov' box in the Initialization Segment has a single track box for that media content component.

   — The track box includes all the different coding information for all the different Representations in the 'stsd' box. Each 'sampleEntry' in the 'stsd' box corresponds to the coding information of the media content component in each Representation. The 'entry_count' in the 'stsd' box shall be equal to the number of different Representations of the media content component.

   — Since a single track box is assigned for a media content component even when there are several differently coded Representations, a single 'track_ID' is used for referencing the media content component in all Representations.

d) For any particular media content component, all track fragments in Media Segments within a same Adaptation Set in the Period shall have the same value of 'track_ID' in 'tfhd' box of 'traf' box of 'moof' box as that of the media content component track in the 'moov' box in the Initialization Segment.

e) The value of 'sample_description_index' in 'tfhd' box in a track fragment of a media content component shall be the index of the corresponding 'sampleEntry' in the 'stsd' box of the media content component track.

f) The 'moof' box shall use movie-fragment relative addressing. Absolute byte-offsets shall not be used. The details are well specified in ISO/IEC 14496-12:2015, 8.8.4 to 8.8.8.

5.2.5.4 Examples

In the following examples, there are two different Representations, say Representation 1 & 2, in an Adaptation Set.

— Representation 1 and Representation 2 have a video and an audio, with the video encoded at 500 kbps and 100 kbps, respectively, and the audio at 96 kbps.

— The total playback duration is 60 sec.

— The playback duration of each Media Segment is 5 sec. Hence each Representation has 12 Media Segments.

— A Media Segment consists of 10 (when a fragment contains both video and audio) or 20 (when a fragment contains a single media content component, i.e., video or audio) movie fragments. Hence, the playback duration of each movie fragment is 0,5 sec.

— The first sample in a fragment is a SAP of Type 1.

— Bitstream switching occurs three times during the 60 sec at 15 sec, 30 sec, and 45 sec as in Figure 17.

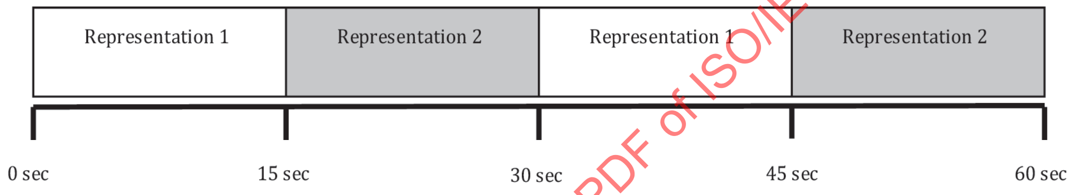


Figure 17 — Example streaming session with bitstream switching occurring at 15th, 30th, and 45th sec

5.2.5.4.1 Example 1

A movie fragment ('moof' box) contains a video track fragment ('traf' box) and an audio track fragment.

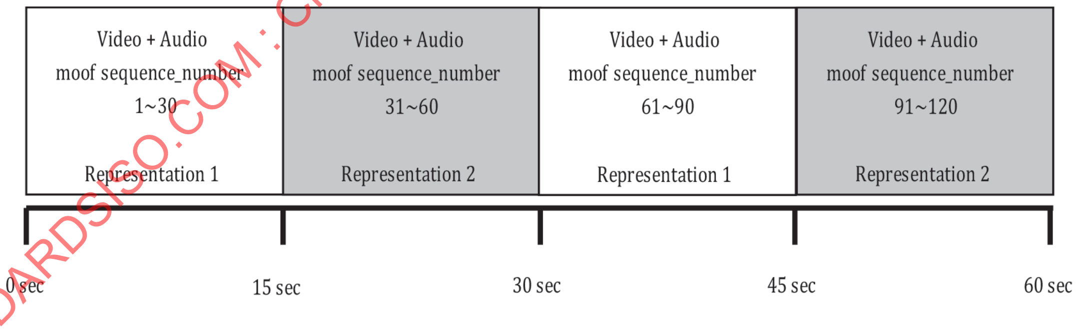


Figure 18 — Streaming session and transmitted Representations/components in Example 1

The concatenated segment file is shown in Figure 19. The following figure shows time relationships of the fragments.

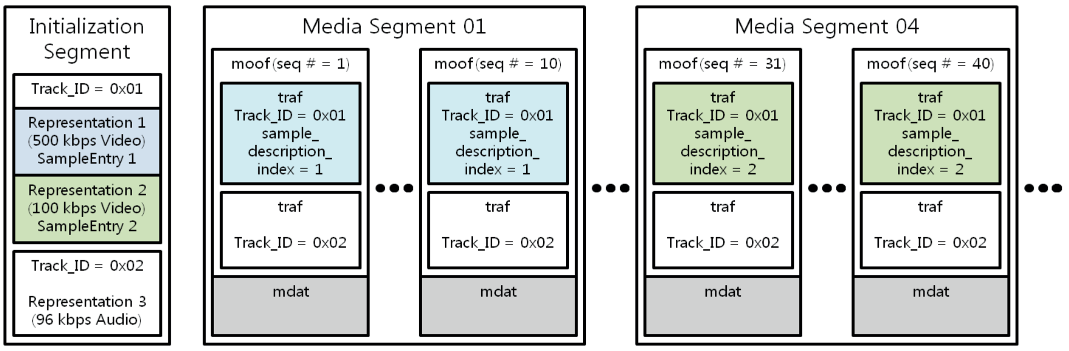

Initialization Segment
Track_ID = 0x01
Representation 1 (500 kbps Video) SampleEntry 1
Representation 2 (100 kbps Video) SampleEntry 2
Track_ID = 0x02
Representation 3 (96 kbps Audio)
Media Segment 01
moof (seq # = 1)
traf Track_ID = 0x01 sample_ description_ index = 1
traf Track_ID = 0x02
mdat
moof (seq # = 10)
Media Segment 04
moof (seq # = 31)
traf Track_ID = 0x01 sample_ description_ index = 2
moof (seq # = 40)

Figure 19 — Concatenated segment file corresponding to Example 1

5.2.5.4.2 Example 2

A fragment contains only a video track fragment or an audio track fragment. The video fragments are interleaved with the audio fragments. Figure 20 shows time relationships of the fragments.

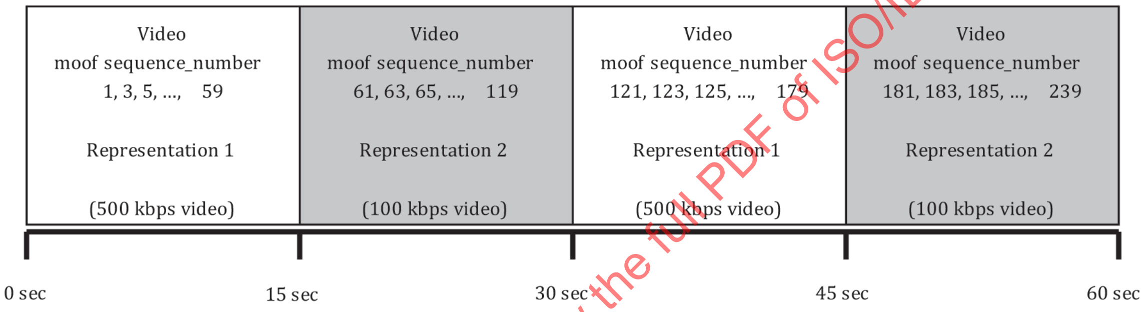

Video moof sequence_number 1, 3, 5, ..., 59
Representation 1
(500 kbps video)
Video moof sequence_number 61, 63, 65, ..., 119
Representation 2
(100 kbps video)
Video moof sequence_number 121, 123, 125, ..., 179
Representation 1
(500 kbps video)
Video moof sequence_number 181, 183, 185, ..., 239
Representation 2
(100 kbps video)
0 sec
15 sec
30 sec
45 sec
60 sec

Figure 20 — Streaming session and transmitted Representations/components in Example 2

The concatenated segment file is shown in the following figure with all the ‘mdat’ boxes are omitted for drawing convenience. Each ‘moof’ box is followed by an ‘mdat’ box that contains the data that is addressed in the ‘moof’ box.

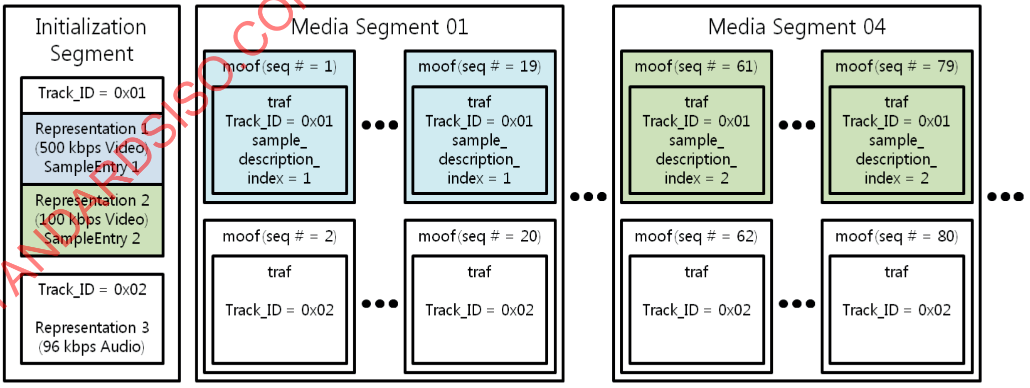


Figure 21 — Concatenated segment file corresponding to Example 2

5.2.5.4.3 Example 3

A movie fragment contains only a video track fragment or an audio track fragment. In a Media Segment, the video fragments are not interleaved with the audio fragments. Figure 22 shows time relationships of the fragments.

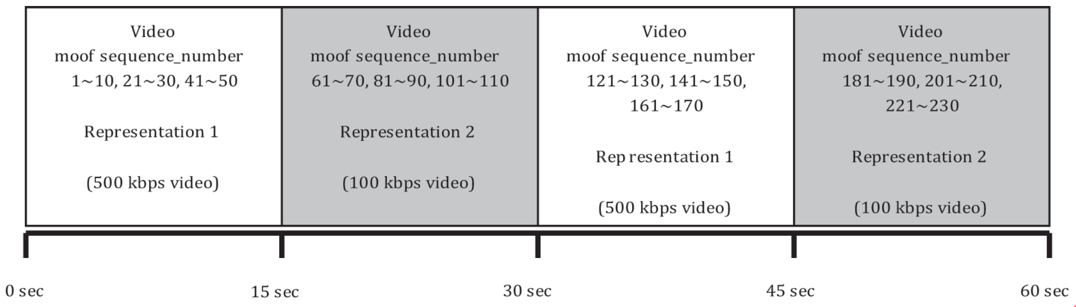


Figure 22 — Streaming session and transmitted Representations/components in Example 3

The concatenated segment file is shown in Figure 23 with all the 'mdat' boxes are omitted for drawing convenience. Each 'moof' box is followed by an 'mdat' box that contains the data that is addressed in the 'moof' box.

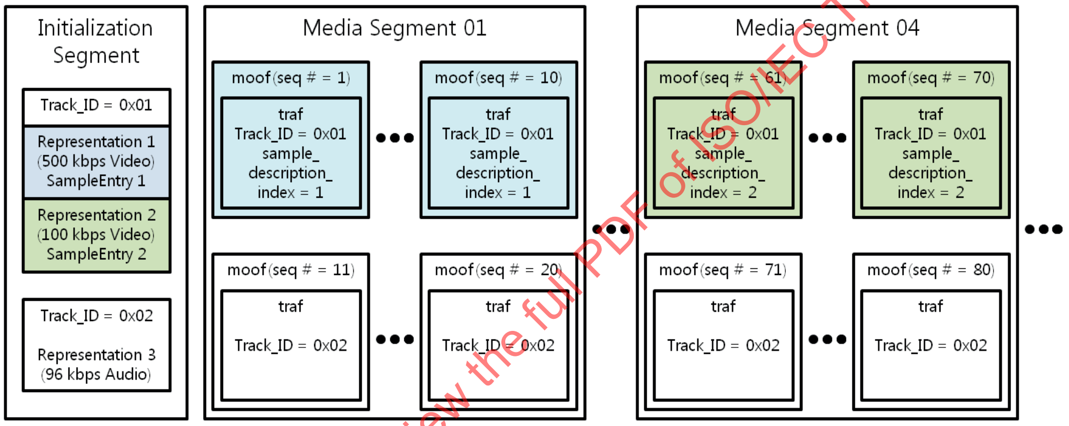

Initialization Segment
Track_ID = 0x01
Representation 1 (500 kbps Video) SampleEntry 1
Representation 2 (100 kbps Video) SampleEntry 2
Track_ID = 0x02
Representation 3 (96 kbps Audio)
Media Segment 01
moof (seq # = 1)
traf Track_ID = 0x01 sample_ description_ index = 1
moof (seq # = 10)
traf Track_ID = 0x01 sample_ description_ index = 1
moof (seq # = 11)
traf Track_ID = 0x02
moof (seq # = 20)
traf Track_ID = 0x02
Media Segment 04
moof (seq # = 61)
traf Track_ID = 0x01 sample_ description_ index = 2
moof (seq # = 70)
traf Track_ID = 0x01 sample_ description_ index = 2
moof (seq # = 71)
traf Track_ID = 0x02
moof (seq # = 80)
traf Track_ID = 0x02

Figure 23 — Concatenated segment file corresponding to Example 3

5.2.5.4.4 Example 4

Same setting as in Example 3, except that each Media Segment is further divided into two Media Segments, one for video and the other for audio. Keeping the audio in separate Media Segments, i.e. in a separate Adaptation Set with a single Representation, saves storage requirements for the DASH server.

The concatenated segment file is shown in Figure 24 with all the 'mdat' boxes are omitted for drawing convenience. Each 'moof' box is followed by an 'mdat' box that contains the data that is addressed in the 'moof' box.

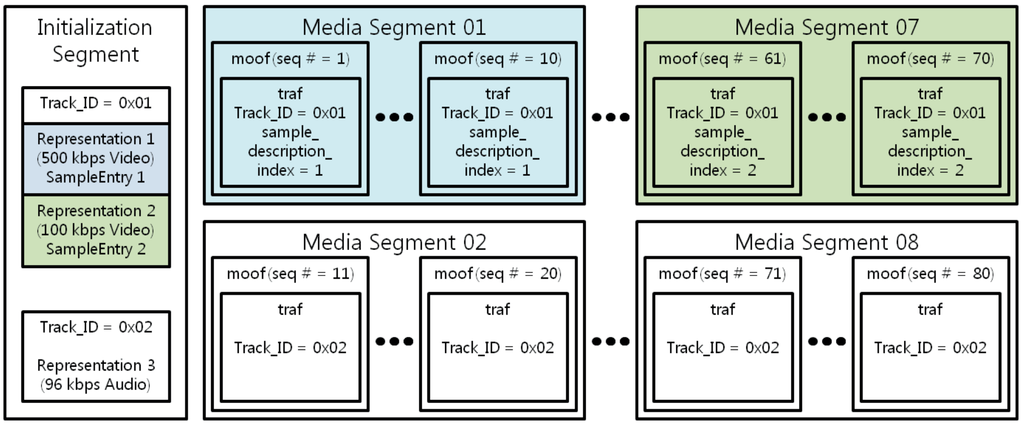


Figure 24 — Concatenated segment file corresponding to Example 4

5.3 Guidelines for MPEG-2 TS content generation

5.3.1 General recommendations

5.3.1.1 General

It is recommended to use MPEG-2 TS Adaptive Profile and adhere to restrictions specified in DASH Simple TS Profile.

5.3.1.2 Media segments

5.3.1.2.1 TS encapsulation

The use of PVR_assist (ETSI 101 154, Annex D) is encouraged, especially for content encrypted using MPEG-2 CA, as this reduces the amount of work required to implement trick modes and makes it feasible for content protected by MPEG-2 CA.

The use of null packets is strongly discouraged, as they serve no purpose in adaptive streaming: constant-bitrate behavior is expected only to the extent the segments have similar sizes.

If PCR PID is a video (or audio) PID, PCR should be present in each TS packet from the PCR PID containing the first byte of the PES header. This is vital for fast startup, random access and switching: after acquiring PAT and PMT, decoders typically drop packets until PCR is encountered. A simple and efficient way of achieving this is use of video PID as PCR PID and carriage of PCR in the first TS packet of the first PES packet of the segment.

5.3.1.2.2 ISO/IEC 14496-10 and ISO/IEC 23008-2

The use of filler NAL units is discouraged as they serve no purpose in adaptive streaming: constant-bitrate behavior is expected only to the extent the segments have similar sizes.

5.3.1.2.3 Bitstream switching segment

Bitstream switching segment should contain the following:

— single TS packet from the video PID, containing a single PES packet, which, in turn, contains only an End Of Sequence NAL unit (for AVC and HEVC) or End Of Sequence header (for MPEG-2 Video);

— a TS packet from the PCR PID containing only adaptation field with discontinuity_indicator flag set to `1`;

— carrying PSI (at least, PAT and PMT is recommended, though not required).

In case of AVC video, it is highly recommended that the concatenation of a media segment and a bitstream switching segment would comply with recommendations of SCTE 172.

5.3.2 Live streaming

5.3.2.1 General

In the live (broadcast) case, real-time encoders are used to simultaneously produce all representations; therefore, the segment duration is critical for minimizing delay. An additional concern is the CDN behavior when a physical file is constantly updated. Therefore, we expect a typical live broadcast deployment to use short segments of roughly equal duration and possibly per-segment index files to allow trick modes on segments within the availability window. The latter would vary, with large windows used for PVR-like functionality.

5.3.2.2 MPD authoring

A dynamic MPD with template segment URL derivation is strongly recommended for live scenarios in order to make MPD generation and updates more efficient and operationally simpler. An example MPD is provided in Figure 25.

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011 DASH-MPD.xsd"
    xmlns="urn:mpeg:DASH:schema:MPD:2011"
    id="a1fd4476-3523-4a1d-99e2-472ae55eb343"
    type="dynamic"
    availabilityStartTime="2012-07-07T07:07:07"
    minBufferTime="PT1.4S"
    profiles="urn:mpeg:dash:profile:mp2t-simple:2011"
    maxSegmentDuration="PT4S"
    minimumUpdatePeriod="PT360S"
    timeShiftBufferDepth="PT240S">

    <BaseURL>http://cdn1.example.com/</BaseURL>
    <BaseURL>http://cdn2.example.com/</BaseURL>

    <Period id="42" >
        <AdaptationSet
            mimeType="video/mp2t" codecs="avc1.4D401F,mp4a" frameRate="30000/1001"
            segmentAlignment="true" bitstreamSwitching="true" startWithSAP="2" >

            <ContentComponent contentType="video" id="481"/>
            <ContentComponent contentType="audio" id="482" lang="en"/>
            <ContentComponent contentType="audio" id="483" lang="es"/>
            <BaseURL>SomeBroadcastProgram_</BaseURL>
            <SegmentTemplate
                  media="$RepresentationID$_$Number%08$.ts"
                  bitstreamSwitching="$RepresentationID$-bssw.ts"
                  duration="4" startNumber="1"/>
            <Representation id="720kbps"  bandwidth="792000"  width="640"  height="368"/>
            <Representation id="1130kbps" bandwidth="1243000" width="704"  height="400"/>
            <Representation id="1400kbps" bandwidth="1540000" width="960"  height="544"/>
            <Representation id="2100kbps" bandwidth="2310000" width="1120" height="640"/>
            <Representation id="2700kbps" bandwidth="2970000" width="1280" height="720"/>
            <Representation id="3400kbps" bandwidth="3740000" width="1280" height="720"/>
        </AdaptationSet>
    </Period>
</MPD>
```

Figure 25 — Example MPD file for live video streaming using MPEG-2 TS

5.3.3 On demand streaming

5.3.3.1 MPD authoring

An example MPD for on-demand streaming using MPEG-2 TS is provided in Figure 26.

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011 DASH-MPD.xsd"
 xmlns="urn:mpeg:DASH:schema:MPD:2011"
 type="static"
 mediaPresentationDuration="PT6158S"
 minBufferTime="PT1.4S"
 profiles="urn:mpeg:dash:profile:mp2t-simple:2011"
 maxSegmentDuration="PT4S">

    <Period id="Movie_01" duration="PT1800S">
        <BaseURL>http://cdn1.example.com/</BaseURL>
        <BaseURL>http://cdn2.example.com/</BaseURL>

            <SegmentTemplate
                media="$RepresentationID$_$Number%05$.ts?poid=88a8c32d8"
                index="$RepresentationID$.sidx"
                bitstreamSwitching="$RepresentationID$-bssw.ts"
                duration="4" startNumber="450"/>

            <Representation id="720kbps"  bandwidth="792000" width="640" height="368" >
                <SubRepresentation level=“0” contentComponent="481" maxPlayoutRate="32"/>
                <SubRepresentation level=“1” contentComponent="481" maxPlayoutRate="4"
                     dependencyLevel=“0”/>
                <SegmentBase timescale="90000" presentationTimeOffset="162000000" />
            </Representation>
            <Representation id="1130kbps" bandwidth="1243000" width="704" height="400">
                <SubRepresentation level=“0” contentComponent="481" maxPlayoutRate="32"/>
                <SubRepresentation level=“1” contentComponent="481" maxPlayoutRate="4"
                     dependencyLevel=“0”/>
                <SegmentBase timescale="90000" presentationTimeOffset="162000000" />
            </Representation>
            <Representation id="1400kbps" bandwidth="1540000" width="960" height="544">
                <SubRepresentation level=“0” contentComponent="481" maxPlayoutRate="32"/>
                <SubRepresentation level=“1” contentComponent="481" maxPlayoutRate="4"
                     dependencyLevel=“0”/>
                <SegmentBase timescale="90000" presentationTimeOffset="162000000" />
            </Representation>
            <Representation id="2100kbps" bandwidth="2310000" width="1120" height="640">
                <SubRepresentation level=“0” contentComponent="481" maxPlayoutRate="32"/>
                <SubRepresentation level=“1” contentComponent="481" maxPlayoutRate="4"
                     dependencyLevel=“0”/>
                <SegmentBase timescale="90000" presentationTimeOffset="162000000" />
            </Representation>
            <Representation id="2700kbps" bandwidth="2970000" width="1280" height="720">
                <SubRepresentation level=“0” contentComponent="481" maxPlayoutRate="32"/>
                <SubRepresentation level=“1” contentComponent="481" maxPlayoutRate="4"
                     dependencyLevel=“0”/>
                <SegmentBase timescale="90000" presentationTimeOffset="162000000" />
            </Representation>
            <Representation id="3400kbps" bandwidth="3740000" width="1280" height="720">
                <SubRepresentation level=“0” contentComponent="481" maxPlayoutRate="32"/>
                <SubRepresentation level=“1” contentComponent="481" maxPlayoutRate="4"
                     dependencyLevel=“0”/>
                <SegmentBase timescale="90000" presentationTimeOffset="162000000" />
            </Representation>
        </AdaptationSet>
    </Period>
</MPD>
```

Figure 26 — Example MPD file for on-demand video streaming using MPEG-2 TS

5.3.3.2 Segment generation

5.3.3.2.1 General

Two main options exist for the VoD case: single period-long media segment and multiple short segments. Both are possible and it is up to the system designer whether to use byte range requests and a single file vs. regular HTTP GET requests and a large amount of small files.

5.3.3.2.2 Media segments

There are no special recommendations for media encoding beyond these specified in Simple TS profile. However, if smooth trick modes are desired, ETSI 101 154, Annex F provides recommendations for that.

5.3.3.2.3 Index segments

In both cases, use of a per-representation hierarchical index file is highly recommended. It is also beneficial to provide Subsegment index (using the `ssix` boxes) in order to allow trick mode access. The latter should index frames, assigning different levels e.g. to I, P, and B frames. An example correspondence of such an index to the video stream structure is illustrated in Figure 27. Note that levels L0 and L1 in the figure correspond to levels 0 and 1 in subrepresentations in the MPD example in Figure 26.

The `ssix` index depicted below is similar to the tier model of ETSI 101 154, Annex D, where level numbers are equivalent to PVR_assist_tier_pic_num field in the PVR_assist structure.

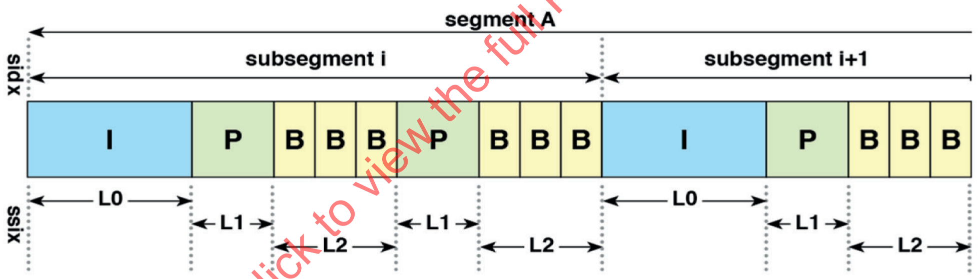


Figure 27 — Example of video coding structure with sub-segments

5.4 Guidelines for Advertisement Insertion

5.4.1 Use cases

Ad markup is typically done at the encoding stage. For *static* ad insertion, ad break locations are known ahead of time. This is the typical case for VoD content.

For content viewed in real-time (e.g. broadcast events at the “live edge”), ad break locations, and durations are only known several seconds ahead of time. This case is referred to as *dynamic* ad insertion.

In terms of architecture, several models are possible. In a *server-driven* model, a generic DASH client is assumed, and all ad decisions, though triggered by the client, are a result of communication between the ad server and the origin server. A *client-driven* model assumes some content and timing parameters being channelled to the client application. The application then communicates with the ad server in order to determine the advertisement content.

5.4.1.1 Ad Decision

Ad decisions are typically made in real time. In case of VoD or pre-recorded content, an ad break can be either taken or skipped, and the break duration may differ. In case of "live edge", ad break should be taken and it has a constant duration.

An ad break can be a sequence of several ads. The composition of an ad break in both static and dynamic case is typically known only in real time.

Ad replacement is a frequently encountered use case. In this workflow, pre-recorded content has original advertisements that were already played (and, hence, have known duration and content), however after some time (e.g. 72 hours) an operator or content provider may replace these ad with different ones.

It is possible that the upcoming (and announced) insertion will be cancelled (e.g. ad break needed to be postponed due to overtime). It is also possible that a planned ad break will need to be cut short, e.g. an ad will be cut short and there will be a switch to breaking news.

5.4.1.2 Ad Representations

Available ad representations can differ greatly from these for the main content in characteristics such as bitrates, resolutions, interlacing, segment duration, codecs, etc. It should be noted that significant differences are undesirable.

Ads are typically unencrypted, even when the content itself is DRM-protected.

5.4.1.3 Trick Modes

Trick mode operations in ad content may be restricted as a matter of business policy. As an example, fast forward may be disallowed on ads.

When same ad break is reached more than once (e.g. as a result of rewind), the ad break may have different composition, duration, or may not be played at all.

5.4.2 Architectures and workflows

5.4.2.1 General

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

Another way of expressing same concept is that server-driven architecture relies solely on DASH native tools, while app-driven approach requires additional non-DASH functionality in order to function.

5.4.2.2 Server-driven architecture

It is recommended to partition the MPD into periods, where each non-remote period consists of either a single ad, or a continuous part of the asset. Ad breaks can be represented by remote periods. As break composition is unknown ahead of time, such remote periods can become after dereferencing several (again, possibly remote) periods. In case of ad replacement, a remote Period element will have content and duration, but still have **`Period`**`@xlink:href` attribute present.

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

When such partition is implemented, **AssetIdentifier** descriptor can be used to mark periods with same content. Periods with identical descriptors indicate that they contain consecutive parts of the same asset. It is highly recommended to use a well-known and unique identification scheme for the purpose.

URLs can be used to embed the relevant ad break information and pass it to the server when a segment or a remote period is requested.

XLink URLs can be used to provide parameters that will be propagated to the ad server in a way opaque to the DASH client. For example, 5.4.2.3 shows embedding an SCTE 35 cue message in an XLink URL as a query parameter.

MPD updates can be used both for real-time ad decision and (in case of dynamic ad insertion) for detecting an expected ad break. Frequent MPD updates can be triggered by setting **MPD**@type="dynamic" and setting a short MPD update interval (e.g. 2 sec). Frequent transmission of complete MPD is highly redundant, as most of the time the MPD does not change; therefore, it is highly recommended to use HTTP conditional GET requests (using If-Modified-Since header, see RFC2616 14.25).

Asynchronous MPD updates are far more efficient for rare events. MPD updates can be triggered by use of inband `emsg` box with MPD Validity Expiration events, MPD Update, or MPD Patch events.

In linear use case, it is possible to use a combination of asynchronous MPD updates and remote periods to achieve just-in-time ad insertion capability: an inband MPD Validity Expiration event triggers an MPD update, and the updated MPD contains a remote period for the upcoming ad break.

In case the main content complies with ISO-BMFF On Demand profile, we can assume it is stored as a single file. There is no need to create per-period files if this content is offered as multi-period. All periods for this asset will have same **BaseURL** values and different **SegmentBase**@presentationTimeOffset values, each corresponding to the media time equivalent to *PeriodStart*. The file will contain `sidx` box(es) and the client will read the index information to calculate the segment offsets taking the value of **SegmentBase**@presentationTimeOffset into account.

Naïve ad tracking and reporting can be done by logging HTTP GET requests coming from the access client.

A more advanced version of ad tracking can be achieved by integrating with the appropriate standard, e.g. IAB VAST [VAST3], and embedding ad server VAST response inside an MPD event. An appropriate external module will process this VAST response and do reporting according to that specification.

5.4.2.3 Example

```
<?xml version="1.0"?>
<MPD xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
    xmlns="urn:mpeg:dash:schema:mpd:2011"
    xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 DASH-MPD.xsd"
    type="dynamic"
    minimumUpdatePeriod="PT2S"
    timeShiftBufferDepth="PT600S"
    minBufferTime="PT2S"
    profiles="urn:mpeg:dash:profile:isoff-live:2011"
    availabilityStartTime="2012-12-25T15:17:50"
    mediaPresentationDuration="PT238806S">
    <BaseURL>http://cdn1.example.com/</BaseURL>
    <BaseURL>http://cdn2.example.com/</BaseURL>
    <!-- Movie -->
    <Period start="PT0.00S" duration="PT1800S" " id="M1">
        <AssetIdentifier schemeIdUri="urn:org:example:asset-id:2013"
            value="md:cid:EIDR:10.5240%2f0EFB-02CD-126E-8092-1E49-W">
        <AdaptationSet mimeType="video/mp4" codecs="avc1.640828" frameRate="30000/1001"
            segmentAlignment="true" startWithSAP="1">
            <BaseURL>video_1/</BaseURL>
            <SegmentTemplate timescale="90000" initialization="$Bandwidth%/init.mp4v"
                media="$Bandwidth$/$Number%05d$.mp4v"/>
            <Representation id="v0" width="320" height="240" bandwidth="250000"/>
            <Representation id="v1" width="640" height="480" bandwidth="500000"/>
            <Representation id="v2" width="960" height="720" bandwidth="1000000"/>
        </AdaptationSet>
    </Period>
    <!-- Mid-roll advertisement, passing base64url-coded SCTE 35 via XLink -->
    <Period start="PT300.00S6S" id="A1"
        xlink:href="https://adserv.com/avail.mpd?acq-timeime=00:10:0054054000&id=1234567&
        sc35cue=DAIAAAAAAAAAAAQAAZ_I0VniQAQAgBDVUVJQAAAAH+cAAAAAA=="
        xlink:actuate="onRequest" >

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

    <!—Movie, cont'd -->
    <Period duration="PT1800S" " id="M2">
        <AssetIdentifier schemeIdUri="urn:org:example:asset-id:2013"
            value="md:cid:EIDR:10.5240%2f0EFB-02CD-126E-8092-1E49-W"/>
        <AdaptationSet mimeType="video/mp4" codecs="avc1.640828" frameRate="30000/1001"
            segmentAlignment="true" startWithSAP="1">
            <BaseURL>video_2/</BaseURL>
            <SegmentTemplate timescale="90000" initialization="$Bandwidth%/init.mp4v"
                media="$Bandwidth%/$Time$.mp4v"/>
            <Representation id="v0" width="320" height="240" bandwidth="250000"/>
            <Representation id="v1" width="640" height="480" bandwidth="500000"/>
            <Representation id="v2" width="960" height="720" bandwidth="1000000"/>
        </AdaptationSet>
    </Period>

</MPD>
```

The example above shows a movie (main content) split in the middle by a 5-min ad break. The ad period is remote, but has default content. In case dereferencing fails, the default content will be played. Ad period(s) coming from a successful dereferencing will be played out otherwise.

Movie periods, *M1* and *M2*, are marked with equivalent `AssetIdentifier` descriptors, hence the player is expected to understand that M2 is a continuation of M1. It would be desirable to try preserving as many resources as possible between M1 and M2 (e.g. initialization segment, etc.). `AssetIdentifier` descriptor uses EIDR in as identification string, allowing unique identification of the asset. This can be used for additional purposes by the application.

Note that it is possible to use `AssetIdentifier` for identifying ads as well; however, one needs to remember that if ad is repeated (e.g. ad periods A1 and AN have same ad), the player will view AN as a continuation of itself. Authors using `AssetIdentifier` for identifying ads are advised to include a string unique for each ad instance (e.g. a GUID or XLink dereferencing response time) in `AssetIdentifier@id`.

5.4.3 App-driven ad insertion

In app-driven ad insertion architecture, user-defined DASH events are used to convey information of upcoming ad break to the app. DASH is used as a transport mechanism, all functionality resulting in insertion resides at the application level.

The DASH access client then downloads the segment, passes the event to some handler provided by the application above the DASH client. An application then requests the ad server which content should be inserted (if at all). The application provides the ad server with parameters passed to it within the SCTE 35 message, possibly accompanied by some client-specific parameters, and asks which content should be played. The ad server responds with instructions what to play (e.g. a new MPD location). A more advanced model would be a dual-client model, where one DASH client is paused and another client plays the ad.

In the VOD case, this functionality can be achieved by use of user-defined MPD events in a static MPD. SCTE 214-1 defines use of SCTE 35 cue messages in an MPD event, and SCTE 214-3 defines carriage of SCTE 35 cue messages in inband events.

For the linear case, let us assume the following architecture: a real-time transcoder at the headend encodes an MPEG-2 TS stream and multicasts the resulting stream within the operator-owned closed network. An edge device, *encapsulator*, then converts continuous streams into DASH segments. The encapsulator also generates MPD and indexes and serves as the origin server for CDN nodes in the georgaphical area it serves.

A real-time transcoder at the operator headend inserts or passes through an SCTE 35 cue message into the encoded MPEG-2 TS, with parameters provided by an ad server. The encapsulator later translates the cue message into a user-defined inband.

5.5 DASH MPD and Segment-based Live Service Offering

5.5.1 Preliminaries

5.5.1.1 MPD Information

In order to offer a service that relies on both, information in the MPD and in Segments, the Service Provider may announce that Segments contains inband information. An MPD as shown in Table 2 provides the relevant information. In contrast to the offering in Table 1, the following information is different.

— The **MPD**`@minimumUpdatePeriod` is present but is recommended to be set to 0 in order to announce instantaneous segment updates.

— The **MPD**`@publishTime` is present in order to identify different versions of MPD instances.

— At least one Representation contains `Representation.InbandEventStream` with `@schemeIDURI` set to `urn:mpeg:dash:event:2012` and the `@value` is 1 or 2.

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

Table 2 — Service Offering with MPD and Segment-based Live Services

MPD Information	Status	Comment
MPD`@type`	mandatory, set to "dynamic"	The type of the Media Presentation is dynamic, i.e. Segments get available over time.
MPD`@publishTime`	mandatory	Specifies the wall-clock time when the MPD was generated and published at the origin server. MPDs with a later value of `@publishTime` shall be an update as defined in 5.4 to MPDs with earlier `@publishTime`.
MPD`@availabilityStartTime`	mandatory	The start time is the anchor for the MPD in wall-clock time. The value is denoted as *AST*.
MPD`@minimumUpdatePeriod`	mandatory	Recommended/mandate to be set to 0 to indicate that frequent DASH events may occur.

Table 2 *(continued)*

MPD Information	Status	Comment
Period@start	mandatory	The start time of the Period relative to the MPD availability start time. The value is denoted as *PS*.
Representation.InbandEventStream	mandatory	If the @schemeIDURI is urn:mpeg:dash:event:2012 and the @value is 1 or 2, then this described an Event Stream that supports extending the validity of the MPD.
SegmentTemplate@media	mandatory	The template for the Media Segment.
SegmentTemplate@startNumber	optional default	Number of the first segment in the Period. The value is denoted as *SSN*.
SegmentTemplate@duration	exactly one of **SegmentTemplate**@duration or **SegmentTemplate.SegmentTimeline** should be present	The duration of each Segment in units of a time. The value divided by the value of @timescale is denoted as *MD*[*k*] with k = 1, 2, ... The segment timeline may contain some gaps.
SegmentTemplate.SegmentTimeline		

5.5.1.2 Segment Information Derivation

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

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

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

5.5.2 Service Offering Requirements and Guidelines

5.5.2.1 Background

In ISO/IEC 23009-1:2014, 5.10, DASH events are defined. For service offerings based on the MPD and segment controlled services, the DASH events specified in ISO/IEC 23009-1:2014, 5.10.4 may be used. Background is provided in the following.

DASH specific events that are of relevance for the DASH client are signalled in the MPD. The URN "urn:mpeg:dash:event:2012" is defined to identify the event scheme defined in Table 3.

Table 3 — InbandEventStream@value attribute for scheme with a value "urn:mpeg:dash:event:2012"

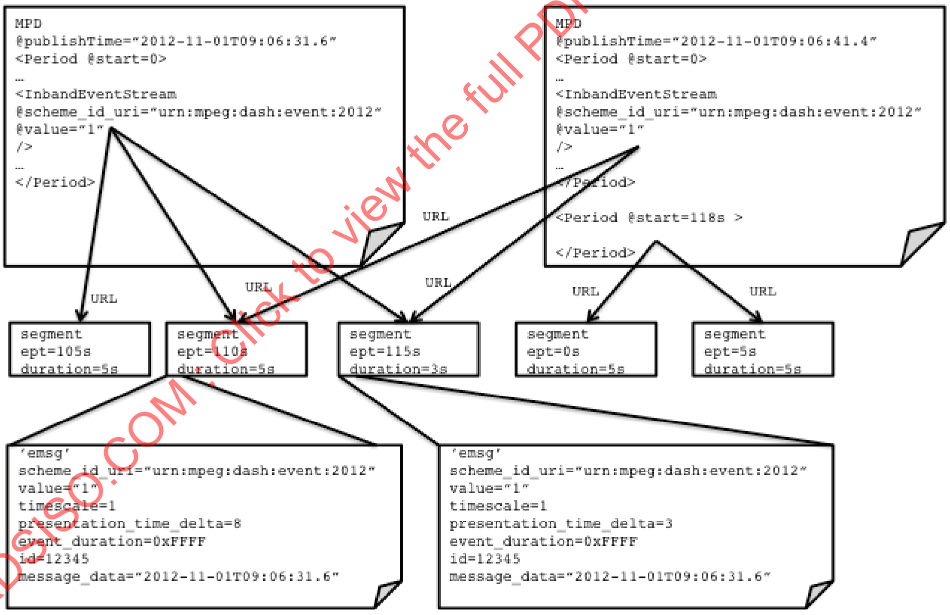
@value	Description
1	Indicates that MPD validity expiration events as defined in ISO/IEC 23009-1:2014, 5.10.4.2 are signalled in the Representation. MPD validity expiration is signalled in the event stream as defined in ISO/IEC 23009-1:2014, 5.10.4.2 at least in the last segment with earliest presentation time smaller than the event time.

Table 3 *(continued)*

`@value`	**Description**
2	Indicates that MPD validity expiration events as defined in ISO/IEC 23009-1:2014, 5.10.4.3 are signalled in the Representation. MPD validity expiration is signalled in the event stream as defined in ISO/IEC 23009-1:2014, 5.10.4.2 at least in the last segment with earliest presentation time smaller than the event time. In addition, the message includes an MPD *Patch* as defined in ISO/IEC 23009-1:2014, 5.10.4.3 in `DASHEvent.mpd_patch` field within the `message_data` field.
3	Indicates that MPD validity expiration events as defined in ISO/IEC 23009-1:2014, 5.10.4.3 are signalled in the Representation. MPD validity expiration is signalled in the event stream as defined in ISO/IEC 23009-1:2014, 5.10.4.2 at least in the last segment with earliest presentation time smaller than the event time. In addition, the message includes a *complete* MPD as defined in ISO/IEC 23009-1:2014, 5.10.4.4 in `DASHEvent.mpd` field within the `message_data` field.

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

Figure 28 shows an example for MPD validity expiration method. An MPD signals the presence of the scheme in one or several Representations. Once a new MPD gets available, that adds new information not present in the MPD with `@publishTime="2012-11-01T09:06:31.6"`, the expiration time of the current MPD is added to the segment by using the emsg box. The information may be present in multiple segments.



```
MPD
@publishTime="2012-11-01T09:06:31.6"
<Period @start=0>
…
<InbandEventStream
@scheme_id_uri="urn:mpeg:dash:event:2012"
@value="1"
/>
…
</Period>
```

```
MPD
@publishTime="2012-11-01T09:06:41.4"
<Period @start=0>
…
<InbandEventStream
@scheme_id_uri="urn:mpeg:dash:event:2012"
@value="1"
/>
…
</Period>

<Period @start=118s >

</Period>
```

```
segment
ept=105s
duration=5s
```

```
segment
ept=110s
duration=5s
```

```
segment
ept=115s
duration=3s
```

```
segment
ept=0s
duration=5s
```

```
segment
ept=5s
duration=5s
```

```
'emsg'
scheme_id_uri="urn:mpeg:dash:event:2012"
value="1"
timescale=1
presentation_time_delta=8
event_duration=0xFFFF
id=12345
message_data="2012-11-01T09:06:31.6"
```

```
'emsg'
scheme_id_uri="urn:mpeg:dash:event:2012"
value="1"
timescale=1
presentation_time_delta=3
event_duration=0xFFFF
id=12345
message_data="2012-11-01T09:06:31.6"
```

Figure 28 — Example for MPD validity expiration to signal new Period

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

— the `message_data` field contains the publish time of an MPD, i.e. the value of the **MPD**`@publishTime`;

— the media presentation time beyond the event time (indicated time by `presentation_time_delta`) is correctly described only by MPDs with publish time greater than indicated value in the `message_data` field;

— the event duration expresses the remaining duration of Media Presentation from the event time. If the event duration is 0, Media Presentation ends at the event time. If 0xFFFF, the media presentation duration is unknown. In the case in which both `presentation_time_delta` and `event_duration` are zero, then the Media Presentation is ended.

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

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

5.5.2.2 Service Offering

A typical service offering with an Inband event stream is provided in Table 4. In this case, the MPD contains information that one or multiple or all Representations contain information that the Representation contains an event message box flow in order to signal MPD validity expirations. The **MPD**@publishTime shall be present.

Table 4 — Basic Service Offering with Inband Events

MPD Information	Value
MPD@type	dynamic
MPD@availabilityStartTime	START
MPD@publishTime	PUBTIME
MPD@minimumUpdatePeriod	MUP
MPD.BaseURL	"http://example.com/"
Period@start	PSTART
InbandEventStream@scheme_id_URI	urn:mpeg:dash:event:2012
InbandEventStream@value	1 or 2
SegmentTemplate@duration	SDURATION

5.5.2.3 Recommendations

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

In this case the following shall apply:

— at least all Representations of all audio Adaptation Sets shall contain an **InbandEventStream** element with `scheme_id_uri="urn:mpeg:dash:event:2012"` and @value either set to 1 or set to 2;

— for each newly published MPD, that includes changes that are not restricted to any of the following (e.g. a new Period):

  — the value of the **MPD**@minimumUpdatePeriod is changed;

  — the value of a **SegmentTimeline.S**@r has changed;

  — a new **SegmentTimeline.S** element is added;

  — changes that do not modify the semantics of the MPD, e.g. data falling out of the timeshift buffer can be removed, changes to service offerings that do not affect the client, etc.

the following shall be done:

— a new MPD shall be published with a new publish time **MPD**@publishTime;

— an 'emsg' box shall be added to each segment of each Representation that contains an **InbandEventStream** element with the following:

  — scheme_id_uri="urn:mpeg:dash:event:2012";

  — DASHEvent structure carrying **MPD**@publishTime and, optionally, patch between the previous and the new MPD (for @value = 2) or full new MPD (for @value = 3);

  — @value to 1..3.

In addition, the following recommendations should be taken into account:

— all Representations of at least one media type/group contain an InbandEventStream element with scheme_id_uri="urn:mpeg:dash:event:2012" and @value either 1, 2, or 3.

5.5.3 Client requirements and guidelines

5.5.3.1 General

A DASH client is guided by the information provided in the MPD. An advanced client model is shown in Figure 29. The advanced client requires parsing of segments in order to determine the following information:

— to expand the Segment List, i.e. to generate the Segment Availability Start Time as well as the URL of the next Segment by parsing the Segment Index;

— to update the MPD based on Inband Event Messages using the 'emsg' box with scheme_id_uri="urn:mpeg:dash:event:2012" and @value either set to 1 or set to 2.

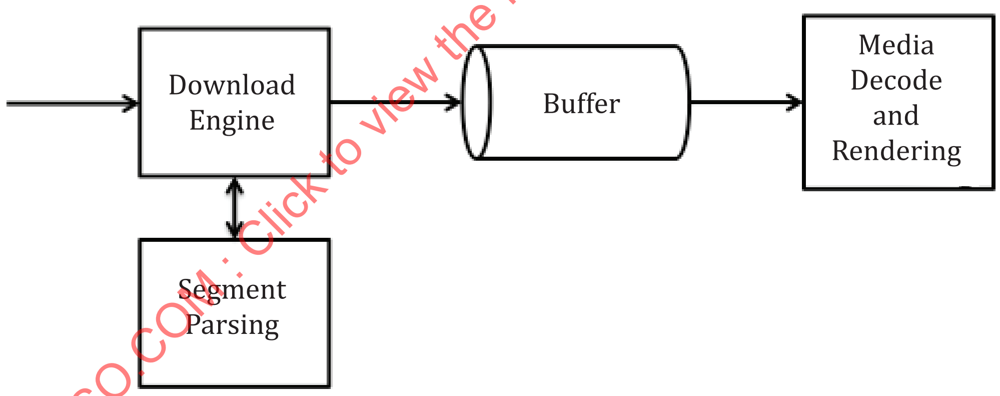


Figure 29 — Advanced client model

Assumes that the client has access to an MPD and the MPD contains the mandatory parameters in Table 4, i.e. it contains the following information:

— **MPD**@minimumUpdatePeriod is set to 0;

— **MPD**@publishTime is included and the value is set to PUBTIME;

— at least on Representation is present that contains an **InbandEventStream** element with scheme_id_uri="urn:mpeg:dash:event:2012" and @value either set to 1 or set to 2;

— either the @duration or **SegmentTimeline** for the Representation is present.

The following example client behavior may provide a continuous streaming experience to the user as documented in the following.

5.5.3.2 MPD Validity expiration and Updates

The DASH client shall download at least one Representation that contains **InbandEventStream** element with `scheme_id_uri="urn:mpeg:dash:event:2012"` and `@value` either set to 1, 2, or 3. It shall parse the segment at least up to the first ‘moof’ box. The DASH client shall parse the segment information and extract the following values:

— `ept` the earliest presentation time of the media segment;

— `dur` the media presentation duration of the media segment.

If an ‘emsg’ is detected, `scheme_id_uri="urn:mpeg:dash:event:2012"` and `@value` either set to 1 or set to 2, the DASH client shall parse the segment information and extract the following values:

— `emsg.publish_time` the publish time documented in the message data of the emsg, either directly or from the patch;

— `emsg.ptd` the presentation time delta as documented in the emsg;

— `emsg.ed` the event duration as documented in the emsg.

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

If no ‘emsg’ validity expiration event is included, then

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

else if an ‘emsg’ validity expiration event is included, then

— the MPD with publish time equal to `emsg.publish_time` can only be used up to a media presentation time `ept + emsg.ptd`. Note that if `dur > emsg.ptd`, then the Period is terminated at `ept + emsg.ptd`,

— any MPD with publish time greater than `emsg.publish_time` can at least be used up to a media presentation time `ept + emsg.ptd`, and

— prior to generating a segment request with earliest presentation time greater than `ept + emsg.ptd`, the MPD shall either

  — be refetched and updated by the client,

  — if `@value=2`, it may patched using the enclosed MPD patch and then updated, or

  — if `@value=3`, it may be updated using the enclosed MPD.

5.5.3.3 Extended Segment Information

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

— `ept` the earliest presentation time of the media segment;

— `dur` the media presentation duration of the media segment.

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

5.6 Guidelines for low latency live service

5.6.1 Use case

Low latency for a live distribution service is essential in various scenarios. One example is the in-venue distribution of an event, such as sports event or a concert. In this case, the delay between the actual live action and the presentation on a mobile device is most appropriate as low as possible in the range of a few seconds at most.

5.6.2 General Approach: Chunked transfer

A general approach to low latency delivery, described in detail in [XXX], is based on chunked transfer as shown in Figure 30. Media segments received by DASH Server are divided into chunks. While a client requests media segment which is partially available, the DASH server should transfer using HTTP/1.1 chunked transfer encoding with the existing chunks and any future chunk(s) as soon as it is ready. Upon receiving chunk(s) of the media segment, DASH client may start to consume the data without waiting for the completeness of the whole segment. Latency could thus be reduced from 1-2 segment durations to 1-2 chunk durations, while still keeping a low start-up delay.

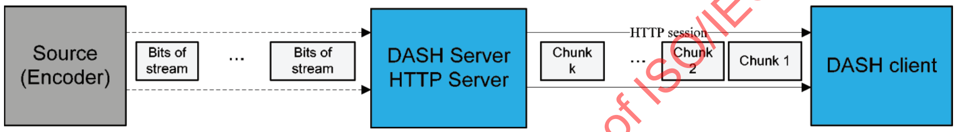


Figure 30 — Chunked Transfer

Table 5 provides an example of the chunked encoding approach. A Media Segment is divided into *n* smaller chunks for chunked encoding. The client requests the Media Segment that is being published (segment *i+1* in line 4). On the server side, although fragment *i+1* is not complete, there are *k* chunks that are already published. In this case, the server would send out chunk *k* immediately after it receives the client request and keep sending the remaining chunks of segment *i+1* once they are ready. The client can play chunk *k* as soon as it is delivered.

Table 5 — Use of chunked transfer encoding

```
1: Client requests bootstrap information;
2: Server responds with bootstrap indicating segment i has been published
3: for (i=0; i < N; i++){ // N is number of segments in this representation
4:   Client requests segment i + 1;
5:   Server checks that chunk k of segment i + 1 has been published;
6:     for (k=0; k≤ n; k++){
7:          // n is the number of chunks in a segment
8:          Server sends out chunk k;
9:          client plays chunk k
10:     }
11: }
```

5.6.3 MPD generation

Despite chunked transfer mode is standard HTTP/1.1 behavior, the mode of operation is supported by additional MPD based signalling. Both **`SegmentBase`** and **`BaseURL`** have attributes named `@availabilityTimeOffset` and `@availabilityTimeComplete`. The former is used to derive the time at which the first byte of a segment is available at the server, while the latter indicates that the media segments of these representations are available completely or only partially at the sender.

Representations for low latency live service should include the `@availabilityTimeComplete` attribute with value "`false`". When the latter is set to "`false`", the `@availabilityTimeOffset` attribute should be present.

6 Client implementation guidelines

6.1 General

A simple example of DASH client behavior can be found in ISO/IEC 23009-1:2014, Annex A. This subclause provides more detailed explanation of client functionality. It also offers recommended practices for implementing DASH clients.

6.2 Client architecture overview

DASH client is a software enabling playback of media content encoded in accordance with ISO/IEC 23009-1. It may be implemented, for example, as

— a stand-alone application,

— a component within the Internet browser or another application,

— a Java-Script embedded in a web-page, or

— an embedded software component in a settop box, TV set, game console, etc.

We show an exemplary architecture of DASH client in Figure 31.

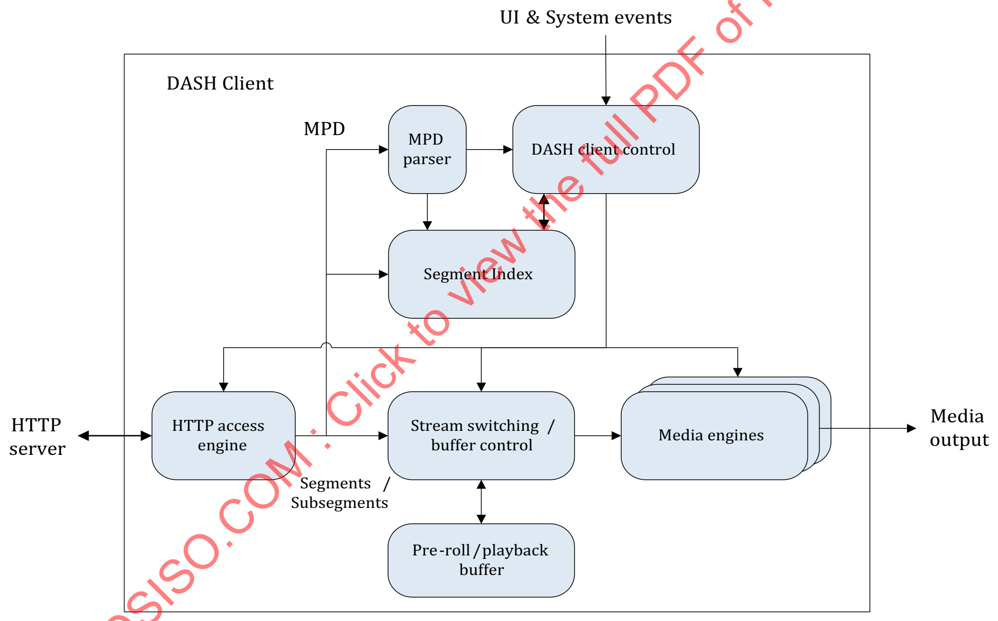


Figure 31 — Example of DASH client architecture

In Figure 31, the client control engine receives user commands, such as “play”, “pause”, or “seek” from an application and translates them into appropriate actions of the DASH client. The HTTP access engine issues requests to HTTP server to receive the Media Presentation Description (MPD), as well as Segments or Subsegments. The MPD parser analyzes the MPD file. The stream switching/buffer control unit receives incoming Segments or Subsegments, places them into a buffer, and schedules them to be delivered to the media playback engine. The actual rendering and playback of multimedia data is accomplished by Media Engines.

In JavaScript, implementations of the DASH client the functions of the “stream switching/buffer control” unit can be delegated to the W3C Media Source Extensions engine.

6.3 Example of client operation

An example of DASH client behavior can be found in ISO/IEC 23009-1:2014, Annex A.

The following subclauses discuss client support of live streaming, MPD retrieval, segment list generation, seeking, support for trick modes, and switching.

6.4 Timing model for live streaming

6.4.1 General

This section offers guidelines for supporting timing model assumed for DASH live streaming services.

6.4.2 MPD information

MPEG-DASH uses a wall-clock time documented in the MPD, which sets up the live Media Presentation. MPEG-DASH assumes that the MPD is generated such that the MPD generation process does have access to an accurate clock. This enables that clients that are synchronized to the wall-clock time to operate closer to the live edge.

Specifically, the following information is available in the MPD when using number-template-based Representations and using the using the `@duration` attribute.

— **`MPD`**`@availabilityStartTime`: The start time is the anchor for the MPD in wall-clock time. The value is denoted as *AST.*

— **`MPD`**`@minimumUpdatePeriod`: The minimum update period of the MPD. The value is denoted as *MUP*.

— **`MPD`**`@suggestedPresentationDelay`: Suggested presentation delay as delta to segment availability start time. The value is denoted as *SPD*.

— **`MPD`**`@minBufferTime`: Minimum buffer time, used in conjunction with the `@bandwidth` attribute of each Representation. The value is denoted as *MBT.*

— **`MPD`**`@timeShiftBufferDepth`: Time shift buffer depth of the media presentation. The value is denoted as *TSB*.

— **`Period`**`@start`: The start time of the Period relative to the MPD availability start time. The value is denoted as *PS*.

— **`SegmentTemplate`**`@startNumber`: Number of the first segment in the Period. The value is denoted as *SSN*.

— **`SegmentTemplate`**`@duration`: The duration of a segment in units of a time. The value divided by the value of `@timescale` is denoted as *d*.

Also, assume that the client did fetch the MPD at fetch time *FT.* Note that a reasonable estimate on the lower value of *FT* is the time when the request for the new MPD is issued and for the higher value *FT* when the MPD is received.

Assuming now that the wall-clock time at the client is denoted at *WT*, and then the client can derive the following information.

— The address of the latest segment that is available on server which requires the latest segment number denoted as *LSN*.

— The segment availability start time of the next segment with number *LSN*+1 and any other segment *SN*, denoted as *SAST*(*SN*). Note that by default, the segment number *SN* starts with 1.

— The media presentation time within the segment that synchronizes closest to the live edge, *MPTL*.

— The media presentation time within the segment that synchronizes to other clients, *MPTS*.

— The time when to fetch a new MPD based on the current presentation time.

Note that the segment availability times are expressing the availability on the origin server.

6.4.3 MPD times

For using the same concept with different timing and addressing schemes, the following two values are introduced according to ISO/IEC 23009-1:

— the position of the segment in the Period denoted as *k* with *k*=1,2,...;

— the MPD start time of the segment at position *k*, referred to as *MST*(*k*);

— the MPD duration of a segment at position *k*, referred to as *MD*(*k*).

Assuming now that the wall-clock time at the client is denoted at *WT*, and then the client can derive the following information:

a) the latest available Period on the server, denoted by its period start time *PS**;

b) the segment availability start time of any segment at position *k* within the Period, denoted as *SAST*(*k*);

c) the position of the latest segment that is available on server in the Period, referred to as *k**;

d) the address of the latest segment that is available on server;

e) the time when to fetch a new MPD based on the current presentation time, or more specifically, the greatest segment position *k'* within this Period that can be constructed by this MPD;

f) the media presentation time within the Representation that synchronizes closest to the live edge, *MPTL*;

g) the media presentation time within the Representation that synchronizes to other clients, *MPTS*.

6.4.4 Context derivation

Using these times, the values from above can be derived as the following:

a) the latest Period is obtained as the Period for which *AST+PS+MD(1) <= NTP*;

b) the segment availability start time is obtained as

$$SAST(k) = AST + PS + MST(k) + MD(k)$$

e) within this Period, the latest segment available on the client is the segment at the position *k** which results in the greatest value for *SAST*(*k**) and at the same time is smaller than *NTP*;

f) the address of the latest segment is obtained by using the position information *k** and then the segment address can be derived. The segment address depends on the addressing method;

g) within this Period, the greatest segment position *k'* that can be constructed by this MPD is the one that results in the greatest value for *SAST*(*k'*) and at the same time is smaller than *FT* + *MUP*.