ISO/IEC JTC 1/SC 29 N

Date: 2014-04-14

ISO/IEC PDTR 23009-3

ISO/IEC JTC 1/SC 29/WG 11

Secretariat:

Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 3: Implementation guidelines

Élément introductif — Élément central — Partie 3: Titre de la partie

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: Technical Report
Document subtype:
Document stage: (30) Committee
Document language: E

STD Version 2.1c2

Contents

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

In exceptional circumstances, the joint technical committee may propose the publication of a Technical Report of one of the following types:

— type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;

— type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;

— type 3, when the joint technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example).

Technical Reports of types 1 and 2 are subject to review within three years of publication, to decide whether they can be transformed into International Standards. Technical Reports of type 3 do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful.

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.

ISO/IEC TR 23009-3, which is a Technical Report of type 3, was prepared by Joint Technical Committee 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 which has been technically revised.

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 23009 provides guidelines for implementation and deployment of streaming media delivery systems based on ISO/IEC 23009 standard. These guidelines include

— guidelines for streaming content generation;

— guidelines for implementation of streaming clients; and

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

Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 3: Implementation guidelines

1 Scope

This part provides technical guidelines for implementing and deploying systems based on ISO/IEC 23009 International Standard.

2 References

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

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

ISO/IEC 23009-2 Information technology — *Dynamic adaptive streaming over HTTP (DASH) — Part 2: Conformance and reference software*.

ISO/IEC 23009-4 Information technology — *Dynamic adaptive streaming over HTTP (DASH) — Part 4: Format independent segment encryption and authentication*.

ITU-T Rec. H.222.0 | ISO/IEC 13818-1, *Information technology – Generic coding of moving pictures and associated audio information: Systems*

ITU-T Rec. H.262 | ISO/IEC 13818-2, *Information technology – Generic coding of moving pictures and associated audio information: Video*

ISO/IEC 13818-3, *Information technology – Generic coding of moving pictures and associated audio information: Audio*

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

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

ISO/IEC 14496-12, *Information technology – Coding of audio-visual objects – Part 12: ISO base media file format (technically identical to ISO/IEC 15444-12)*

ITU-T Rec. H.265 | ISO/IEC 23008-2, *Information technology – Coding of audio-visual objects – Part 2: High Efficiency Video Coding*

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 23001-7, *Information technology – MPEG systems technology – Part 7: Common encryption in ISO base media file format files*

ISO/IEC 23001-8, *Information technology – MPEG systems technologies – Part 8: Coding-independent code points*

IETF RFC 1521, *MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*, September 1993

IETF RFC 1738, *Uniform Resource Locators (URL)*, December 1994

IETF RFC 2141, *URN Syntax*, May 1997

IETF RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1*, June 1999

IETF RFC 3023, *XML Media Types*, January 2001

IETF RFC 3406, *Uniform Resource Names (URN) Namespace Definition Mechanisms*, October 2002

IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*, January 2005

IETF RFC 4122, *A Universally Unique IDentifier (UUID) URN Namespace*, July 2005

IETF RFC 4337, *MIME Type Registration for MPEG-4*, March 2006

IETF RFC 5646, *Tags for Identifying Languages*, September 2009

IETF RFC 6381, *The 'Codecs' and 'Profiles' Parameters for "Bucket" Media Types*, August 2011

W3C XLINK *XML Linking Language (XLink) Version 1.1*, W3C Recommendation 06, May 2010

ETSI TS 101 154, *Digital Video Broadcasting (DVB); Implementation guidelines for the use of Video and Audio Coding in Broadcasting Applications based on the MPEG-2 Transport Stream*, September, 2009.

SCTE 172, *Constraints on AVC Video Coding for Digital Program Insertion,* 2011.

W3C, *Media Source Extensions*, W3C Recommendation (Draft), 18, January 2013.

W3C, *Encrypted Media Extensions*, W3C Recommendation (Draft), 22 January 2013.

3 Terms, Definitions and Abbreviated Terms

This document uses definitions, symbols, and abbreviated terms defined in ISO/IEC 23009-1.

Additionally, this document uses 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.

Additionally, this document uses audio coding terms defined in ISO/IEC 13818-1, ISO/IEC 14496-3, ISO/IEC 23003-1, and ISO/IEC 23003-3.

4 Introduction

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.

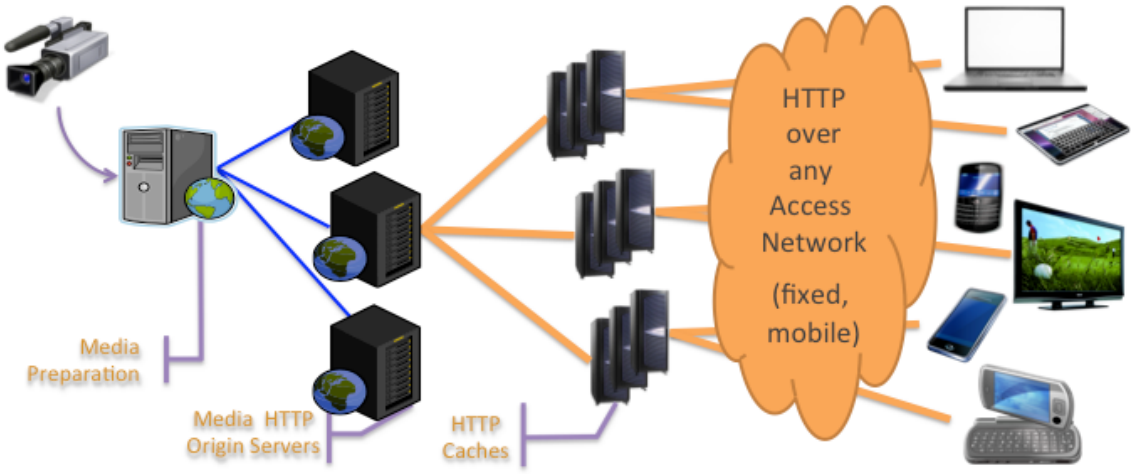

Figure 1 – Example DASH-based Media Distribution Architecture.

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.

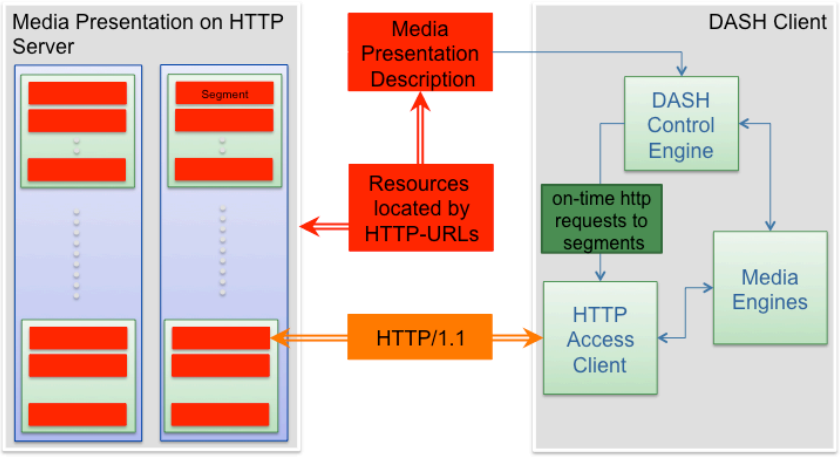

Figure 2 – Standardized aspects in DASH. Normative components are marked in red.

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.

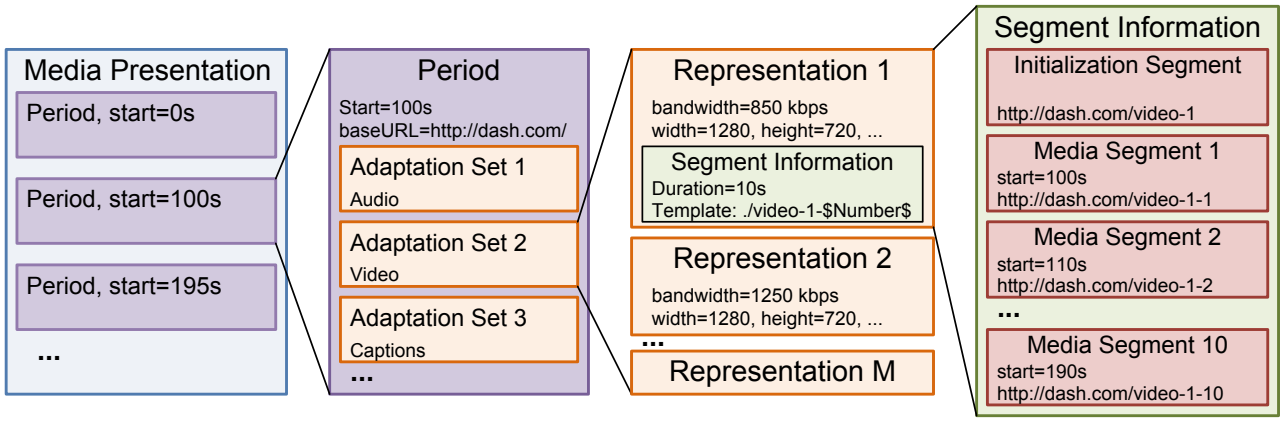

Figure 3 – DASH hierarchical data model.

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, 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 must 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, and
- 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, clause 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;
- *SegmentTemplate* element, providing a template form of URL(s) for Media Segments.

Details of these schemes are described in ISO/IEC 23009-1, Clause 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:

- 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, 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 Clause **Error! Reference source not found.** of this document.

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.

ISO/IEC 23009-1 defines six profiles, shown in Figure 4. 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 file formats. These profiles do not impose restrictions on codecs encapsulated by these file formats.

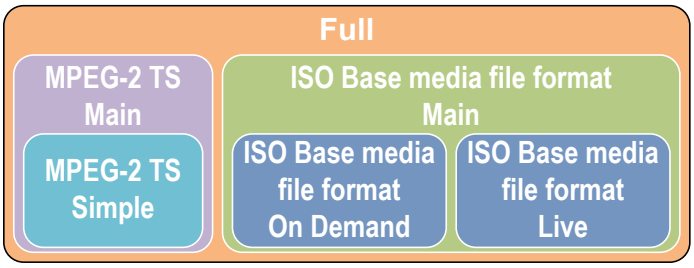

Figure 4 – DASH Profiles.

In addition to six profiles defined in ISO/IEC 23009-1, additional profiles may also be defined externally, by means of Registration Authority. Such profiles may also impose constrains on codecs, resolutions, bit-rates, segment durations, and other system parameters.

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 100Kbps to 3Mbps, a reasonable set of operating points may include: 100kbps, 200kbps, 400kbps, 600kbps, 1.2Mbps, 1.8Mbps and 2.5Mbps,

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 5.

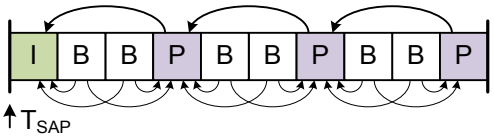

Figure 5 – Type 1 SAP implemented using Closed GOP structure.

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 6.

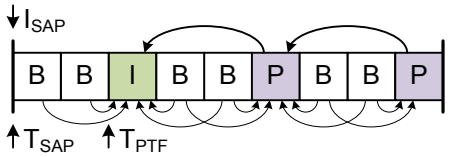

Figure 6 – Type 2 SAP implemented using modified Closed GOP structure.

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 7.

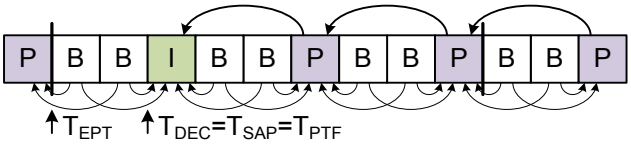

Figure 7 – Type 3 SAP implemented using Open GoP structure.

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 must be prepared using identical codecs, resolutions, and prediction structures (codecs must 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 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 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 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 must 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. Since it is based on an AAC-LC core coder, all restrictions for AAC-LC should also apply for HE-AAC. Additional adaptations for the AAC-LC core are needed to avoid dropouts 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

The SBR decoder analysis framing is delayed by 6 time slots (6*64 samples) compared to the framing of the AAC-LC core decoder. In addition the analysis QMF adds another 320 samples delay. 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 accordingly.

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 clause 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 a 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-LS / 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 restrictionss 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 must 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 one.

The MPEG Surround tool *residual coding* employs a representation of differential signals using the AAC-LC syntax. As described in Clause **Error! Reference source not found.**, 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 Clause 6.4 of this specification.

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 5.1.3.2.1 below)
M	`MPD@id`	(1)
I	`Period@id`	(1)
A	`AdaptationSet@id`	(1)
g	`AdaptationSet@group`	
l	`AdaptationSet@lang`	(3)
P	`AdaptationSet@par`	(2)
R	`Representation@id`	(1)
B	`Representation@bandwidth`	
W	`Common@width`	
H	`Common@height`	
s	`Common@sar`	(2)
F	`Common@framerate`	(2)
F	`Common@audioSamplingRate`	(2)
N	`$Number$`	(4),(5)
T	`$Time$`	(5)
C	`Common@codecs`	(2)

5.1.4.2.1 Restrictions and processing rules

The following restrictions and processing rules should apply.

1. **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.
2. Characters '/' , ':', and '.',e.g. in @framerate, @sar, @codecs, should be replaced with '-'. Thus, framerate of 24000/1001 should be written as _f24000-1001.
3. Whitespace-separated lists should be written as two separate fields, e.g. for lang="en es" the filename would include "_len_les"
4. Segment Number should be zero-padded to width of at least 5 digits, i.e., it should appear e.g. as $Number%05$. The reasoning behind this restriction is to make sure lexical ordering and numerical ordering are identical for the whole Representation;
5. 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 .
6. 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_$Number$
7. **Common**@bandwidth should be expressed in kilobits per second, and end with the character "k".
8. Naming is case-insensitive.
9. 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 Clause 5.1.3.2 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 8 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"
    duration="4"
```

```
    startNumber="1"/>

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

Figure 8 – 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 Clauses 8.1, 8.3.1, and 8.3.2 of ISO/IEC 23009-1.

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 9. This MPD document describes content available from two sources (cdn1 and cdn2) that has audio available in English or French at rates of 64kbits and 32kbits and subtitles in German. Six versions of the video are provided at bitrates between 256kbit/s and 2Mbit/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 9 – 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 Clauses 8.1, 8.3.1, and 8.3.3 of ISO/IEC 23009-1.

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 Clauses 5.3.9.5.2 and 5.3.9.5.5 of ISO/IEC 23009-1 correspondingly.

Figure 10 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.

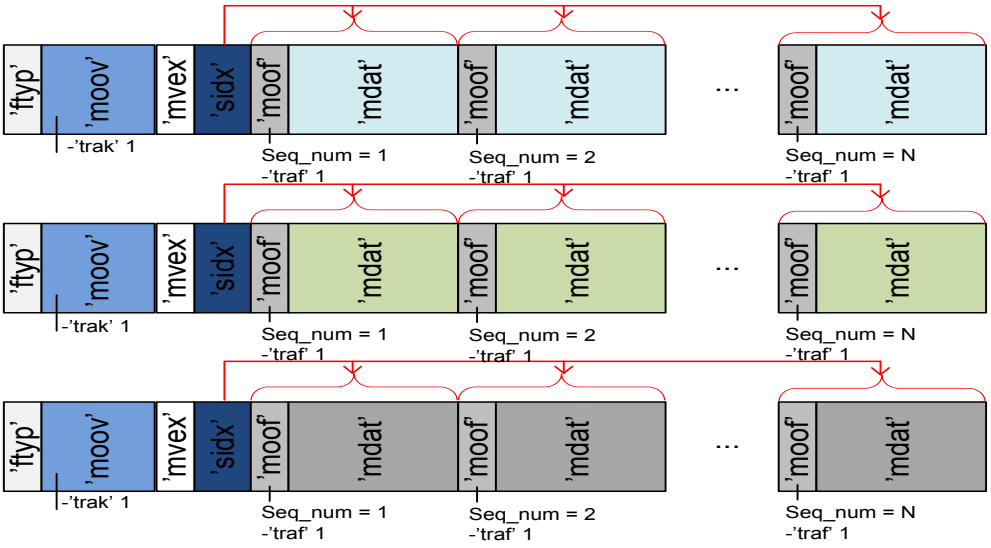

Figure 10 – Example set of Segments for on-demand video streaming using ISOMBFF.

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 Clauses 8.1, 8.3.1, and 8.3.2 of ISO/IEC 23009-1.

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 Clauses 8.1, 8.3.1, and 8.3.3 of ISO/IEC 23009-1.

In addition, segments encoded as dependent representations should fulfill the following constraints:

(1) The Segments in the dependent Representations are simple Indexed Media Segments as defined in ISO/IEC 23009-1, Clause 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, Clause 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.

(2) 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, Clause 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.

(3) 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.

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

$$Lmf(S_i, R_n) < Fmf(S_i, R_j) \quad \forall i \cap \forall n \,|\, 0 \le n < j \le N \qquad \text{eq. (1)}$$

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 11 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.

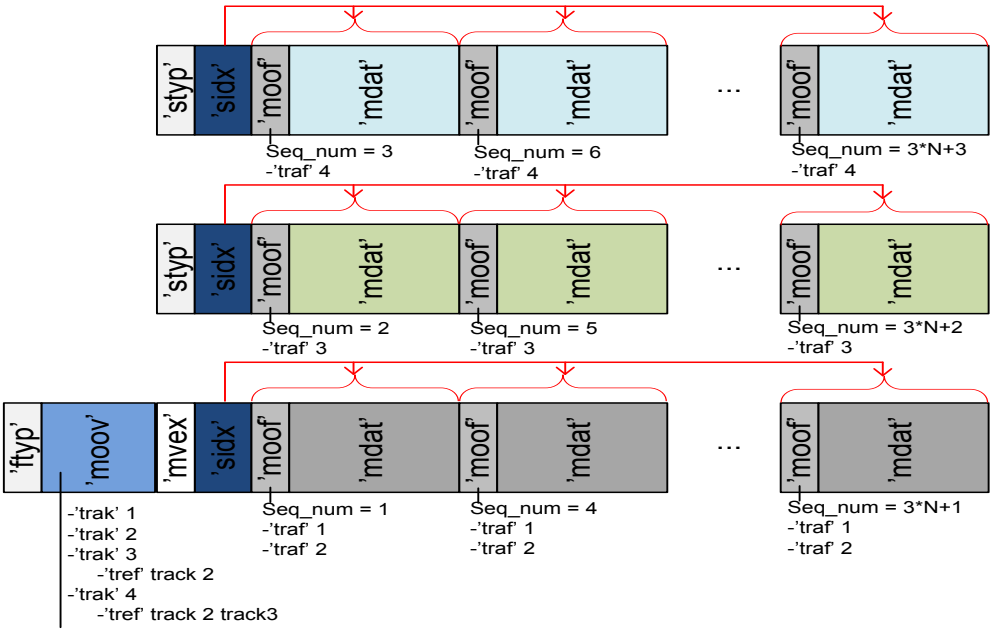

Figure 11 – Segments for an Adaptation Set containing dependent Representations

As shown in the figure, 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 eq.(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 Clauses 8.1, 8.4.1, and 8.4.2 of ISO/IEC 23009-1.

Clause **Error! Reference source not found.** of this specification 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 Clauses 8.1, 8.4.1, and 8.4.3 of ISO/IEC 23009-1.

Segment generation process should be arranged such that it is possible to implement timely MPD updates and follow time model described in Clause **Error! Reference source not found.** of this specification.

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 must providing support for Live services and enable seamless switching. The 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 Clauses 8.1, 8.4.1, and 8.4.2 of ISO/IEC 23009-1.

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 Clauses 8.1, 8.4.1, and 8.4.3 of ISO/IEC 23009-1.

In additions, segments encoded as dependent representations shall fulfill the following constraints:

(1) 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.
(2) The sequence_number in the ‘mfhd’ of the movie fragments in the segments fulfil eq. (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 an 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 Clauses 8.1, 8.3.1, and 8.3.2 of ISO/IEC 23009-1.

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 Clauses 8.1, 8.3.1, and 8.3.3 of ISO/IEC 23009-1.

In addition, the following conditions should be satisfied:

(1) The Initialization Segment should contain the Level Assignment (‘`leva`’) box with the same levels as provided in SubRepresentation@level.
(2) All Media Segments should conform to Sub-Indexed Media Segments as defined in ISO/IEC 23009-1, Clause 6.3.4.4 and therefore should include ‘sims’ as compatible brand in the ‘styp’ box.
(3) 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.
(4) 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.
(5) 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.
(6) 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.
(7) 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. (3), (4), (5) and (6).

When (3) 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 (6), it is necessary to arrange all the samples belonging to each of the leves at the beginning of the Subsegment. As an example Figure 12 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 $IB_1B_0B_1P\ldots$ in presentation order.

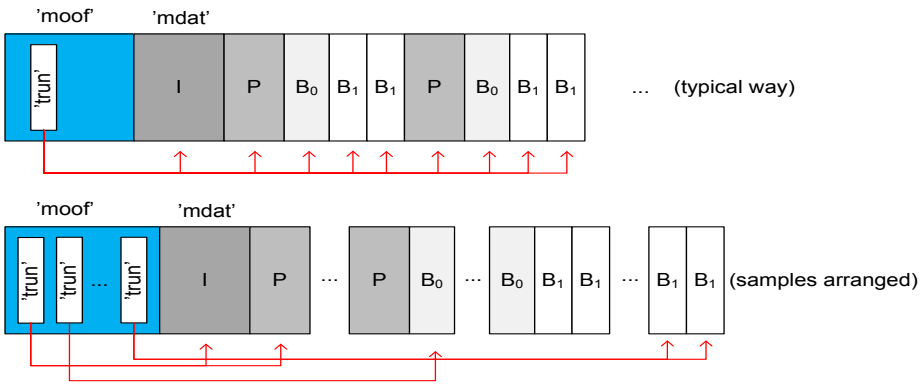

Figure 12 – Movie fragment format for arranged samples for easing fast forward with ‘ssix’ box.

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 (4) of (5) 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 Section **Error! Reference source not found.** 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 Section **Error! Reference source not found.**) are a particular use case of sub-representations for a specific purpose.

5.2.4.2 MPD authoring

Same rules as described in Section 5.2.3.2 apply.

5.2.4.3 Segment generation

Same rules as described in Section 5.2.3.3 apply.

In Figure 13, 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).

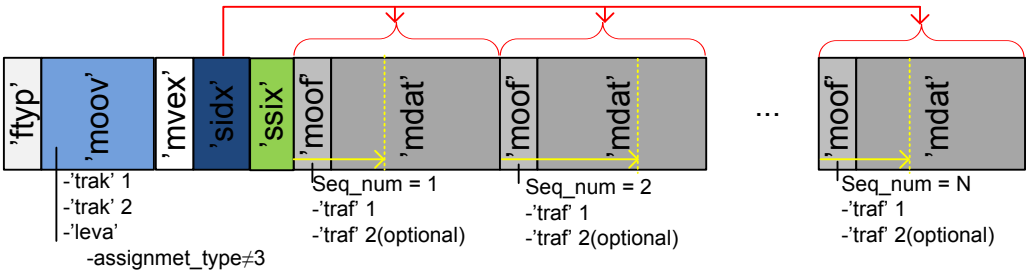

Figure 13 – Example of usage of ‘ssix’ box for Sub-Representations for self-initialising segment with assignment type in ‘leva’ box other than 3.

In Figure 14, 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.

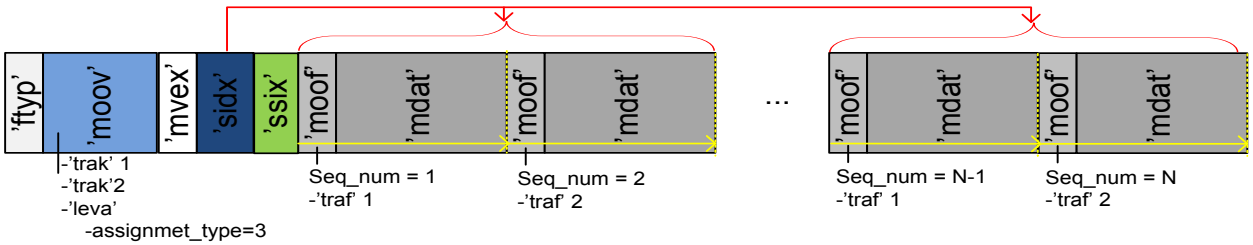

Figure 14 – Example of usage of 'ssix' box for Sub-Representations for self-initialising segment with assignments type in 'leva' box equal to 3.

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:

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

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

(3) 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.

(4) 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.

(5) 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.

(6) The ‘moof’ box shall use movie-fragment relative addressing. Absolute byte-offsets shall not be used. The details are well specified in 8.8.4~8.8.8 of ISO/IEC 14496-12.

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, 30, and 45 sec as in the following figure.

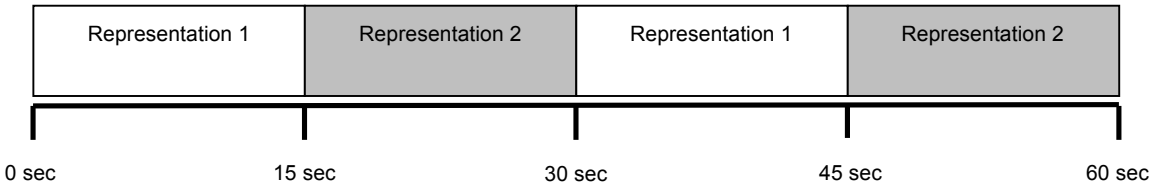

Figure 15 – An example streaming session with bitstream switching occurring at 15th, 30th, and 45th sec.

5.2.5.4.1 Example 1

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

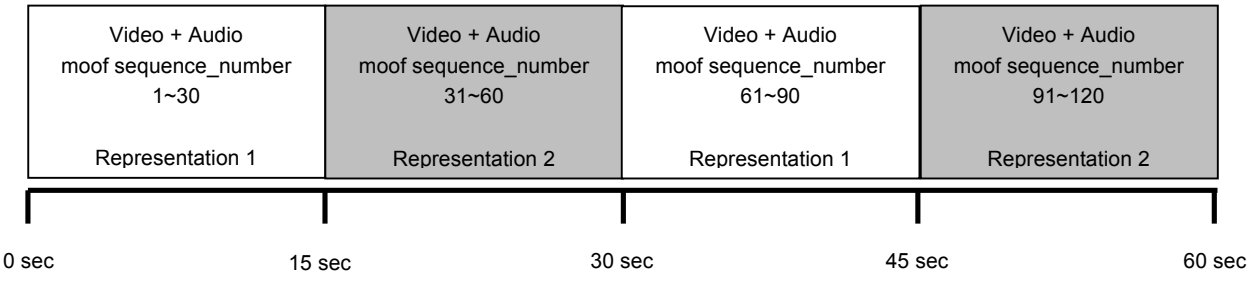

Figure 16 – Streaming session and transmitted Representations / components in Example 1.

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

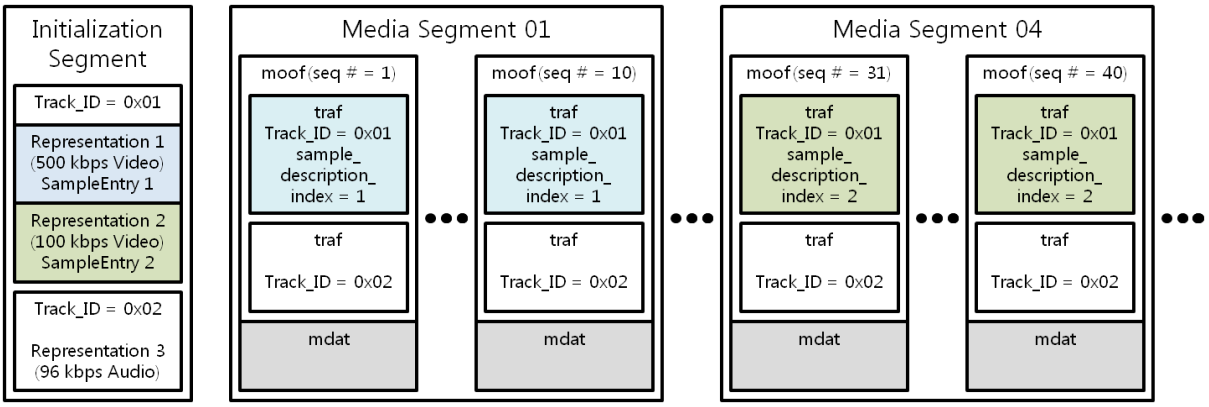

Figure 17 – Concatenated segment file corresponding to Example 1.

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. The following figure shows time relationships of the fragments.

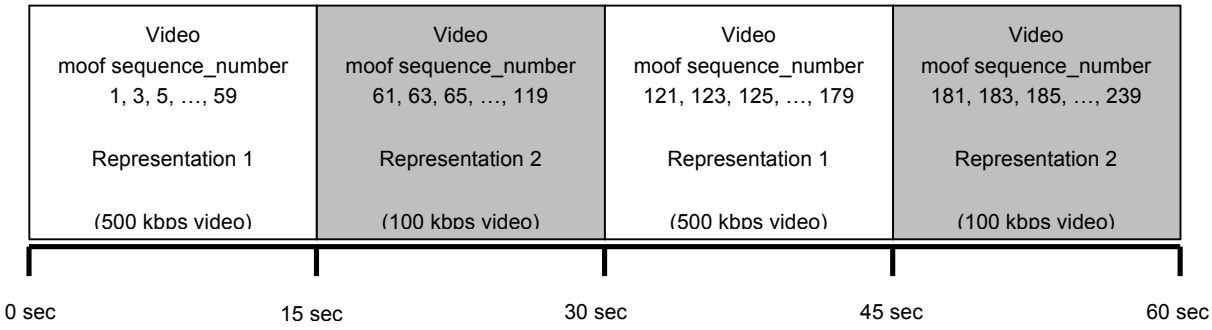

Figure 18 – Streaming session and transmitted Representations / components in Example 2.

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.

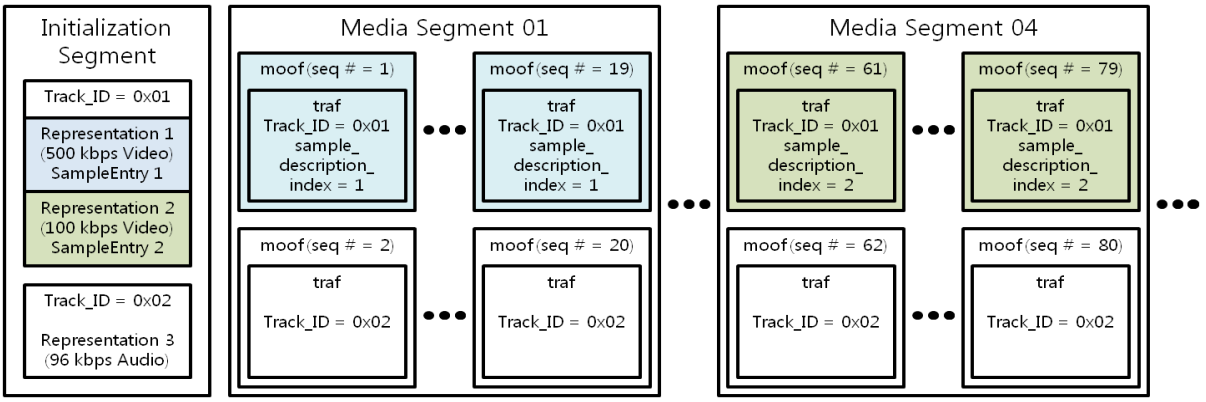

Figure 19 – Concatenated segment file corresponding to Example 2.

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. The following figure shows time relationships of the fragments.

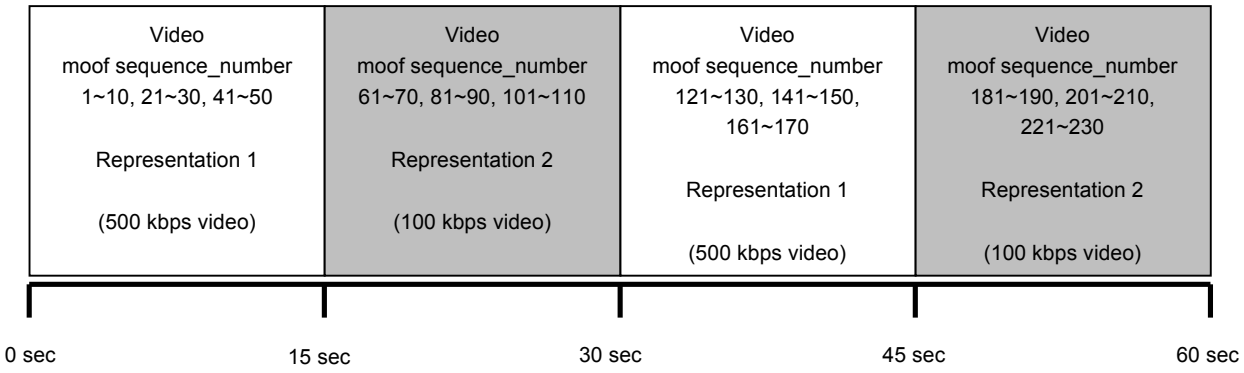

Figure 20 – Streaming session and transmitted Representations / components in Example 3.

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.

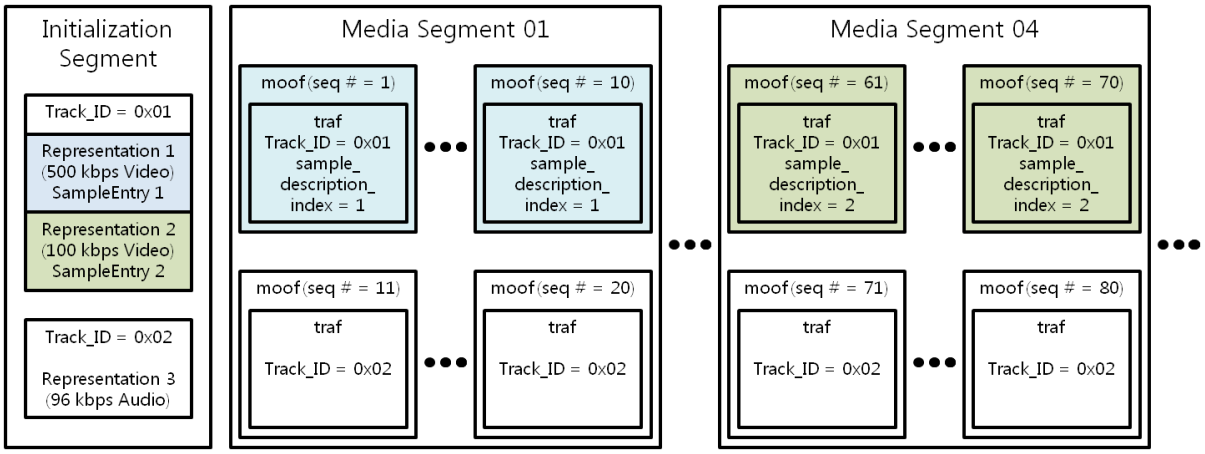

Figure 21 – Concatenated segment file corresponding to Example 3.

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 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.

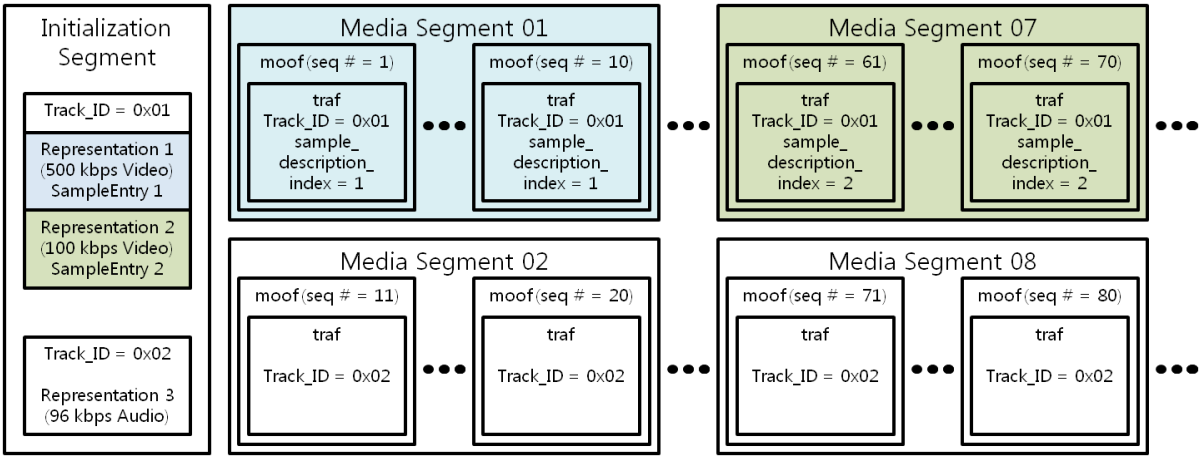

Figure 22 – Concatenated segment file corresponding to Example 4.

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 contetn 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 untill PCR is encountered.

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`.
- PSI (at least – PAT and PMT), in case media segments are not self-initializing.

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 23.

```
<?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 23 – 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 24 – 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 media segment, and multiple short segments, as described in Clause **Error! Reference source not found.**. 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 subclause **Error! Reference source not found.**.

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.

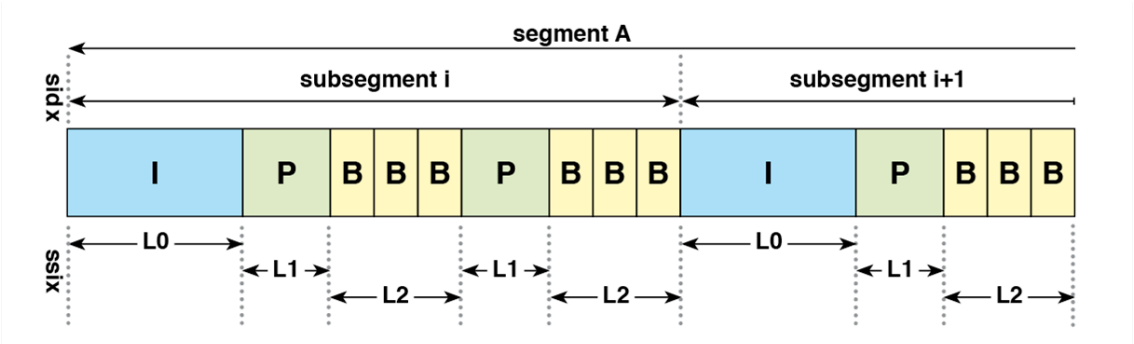

Figure 25 – Example of video coding structure with sub-segments.

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 (e.g., 4 sec.). The latter 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 channeled 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 must 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.

It is possible that the upcoming (and announced) insertion will be canceled (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 MPD authoring

It is recommended to partition the MPD into periods, where each period consists of either a single ad, or a continuous part of the asset. In most cases, this partition will be necessary (e.g., due to different representations, content protection, remote periods, etc.).

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.

Remote periods, using XLink for just-in-time MPD assembly, are a very convenient way of implementing ad insertion.

When the composition of an ad break is unknown ahead of time (i.e., number of ads or/and their duration will be decided in real time), remote periods with zero duration can be utilized. It is possible to include more than one element of the same type in a remote element, hence a single zero-duration remote period may resolve into multiple periods each carrying a single advertisement.

Note that XLink support is not required for most DASH profiles.

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 MPD updates are inefficient; as most of the time the MPD does not change it is highly recommended to use HTTP conditional requests.

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

5.4.3 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">
        <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 -->
    <Period xlink:href="http://adserver.com/movie_11/ad_1.mpd" xlink:actuate="onRequest"
        duration="0S"/>
    <!—Movie, cont'd -->
    <Period duration="PT1800S">
        <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>
```

5.4.4 Use of inband events

5.4.4.1 General

DASH Inband Events are a powerful tool that can allow client-driven or client-assisted ad insertion. The main capability added by events is ability to pass opaque information to a DASH client or/and the application layer above in a manner that keeps this information tightly synchronized with the media timeline.

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

NOTE: Inband events should be used for purposes that are related to media and are meaningful e.g. when consumed in non-realtime (VoD, nPVR type of applications).

5.4.5 Client-driven ad insertion

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 segments which cab be of several strains, e.g., DASH/ISO-BMFF with Common Encryption (ISO/IEC 23001-7) in both AES-CTR and AES-CBC modes, as well as DASH/TS + Segment Encryption (ISO/IEC 23009-4). 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 an SCTE 35 splice_insert() command into the encoded MPEG-2 TS, with parameters provided by an ad server.

The encapsulator later re-encapsulates the SCTE 35 command into the payload of the `emsg` box and adds it to the beginning of the segment

The DASH access client then downloads the segment, parses the `emsg` box, recognizes the event scheme and passes the payload of the event to some handler provided by the application above the DASH client.

An application then asks the ad server which content should it switch to (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 one DASH client is paused and another client plays the ad.

Example G.9 in ISO/IEC 23009-1 shows the `emsg` box used for passing an XML-ized representation of SCTE 35 segmentation descriptor to the client application. This would be an example of such use of `emsg` box with a user-defined payload.

5.5 Guidelines for Low Latency Live Service

5.5.1 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.

5.5.2 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 5.5.1. 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.

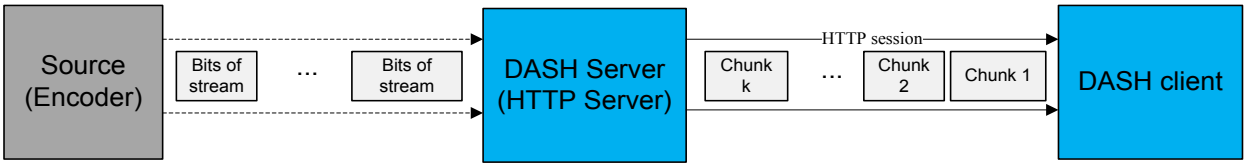

Figure 5.5.1 Chunked Transfer

Table 5.5.1 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 5.5.1: 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: }
```

5.5.3 MPD generation

Despite chunked transfer mode is standard HTTP/1.1 behaviour, 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 Annex A of ISO/IEC 23009-1. This section 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, or
- a Java-Script embedded in a web-page,
- an embedded software component in a settop box, TV set, game console, etc.

We show an exemplary architecture of DASH client in Figure 26 – Example of DASH client architecture.Figure 26.

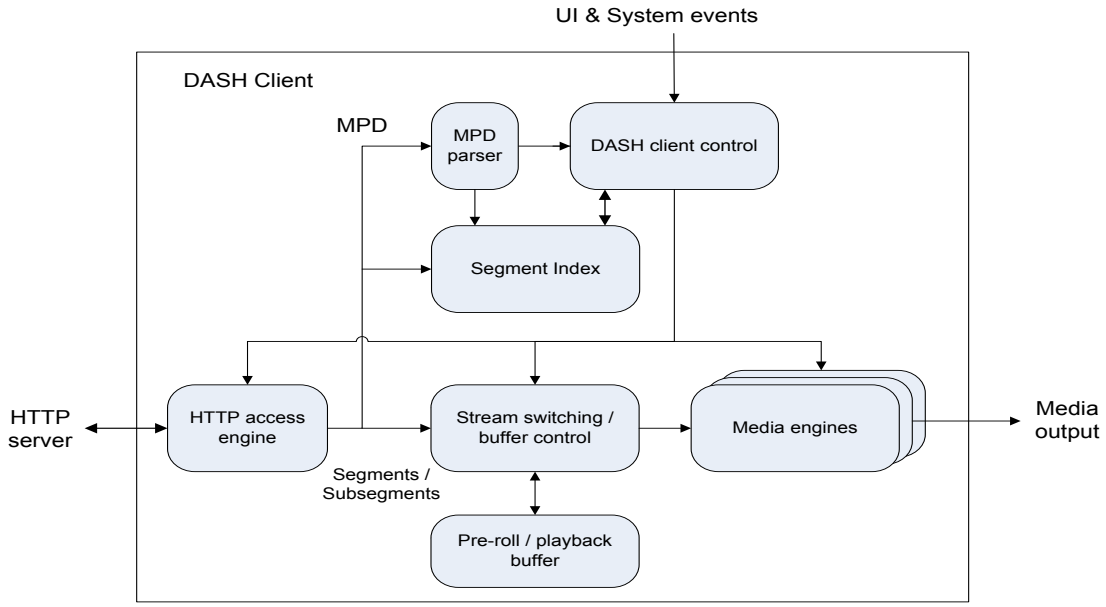

Figure 26 – Example of DASH client architecture.

In this Figure, 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 Annex A.

The following clauses 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 then 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:

1. the latest available Period on the server, denoted by its period start time *PS**
2. The segment availability start time of any segment at position *k* within the Period, denoted as *SAST*(*k*).
3. The position of the latest segment that is available on server in the Period, referred to as *k**
4. The address of the latest segment that is available on server
5. 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.
6. The media presentation time within the Representation that synchronizes closest to the live edge, *MPTL*.
7. 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:

1. The latest Period is obtained as the Period for which *AST+PS+MD(1)* <= *NTP*.
2. The segment availability start time is obtained as
$$SAST(k) = AST + PS + MST(k) + MD(k)$$
3. 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*.
4. 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.
5. 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*.

6.4.5 Derivation of MPD times

6.4.5.1 Duration attribute

If the `@duration` attribute is present and the value divided by the value of `@timescale` is denoted as *d* then the MPD times are derived as:

- *MD*(*k*) = *d*
- *MST*(*k*) = (*k*-1)**d*

6.4.5.2 Usage of segment timeline

In case the Segment base information contains a **`SegmentTimeline`** element with N_s **S** elements indexed with s=1, ..., N_s, then

- the *t*[*s*] is the value of `@t` of the *s*-th **S** element divided by the value of the `@timescale` attribute,
- the *d*[*s*] is the value of `@d` of the *s*-th **S** element divided by the value of the `@timescale` attribute,
- the *r*[*s*] is the value of `@r` of the *s*-th **S** element (unless the `@r` value is -1, which means that the value is unknown and the `@d` may be used until updated information is available)

Then MPD duration and start times can be derived as follows:

- *k*=0
- for *s=1, ...* N_s
    - *k* = *k* + 1
    - *MST*(*k*) = *t*[*s*]
    - *MD*(*k*) = *d*[*s*]
    - for *j* = 1, ..., *r*[*s*]
        - *k* = *k* + 1
        - *MST*(*k*) = *MST*(*k-1*) + *d*[*s*]
        - *MD*(*k*) = *d*[*s*]

6.4.6 Addressing methods

6.4.6.1 Introduction

The addressing method is independent of the usage of the timeline generation. The interpretation of the `@startNumber` depends on the addressing method.

6.4.6.2 Playlist method

If the Representation contains or inherits one or more **`SegmentList`** elements, providing a set of explicit URL(s) for Media Segments, then the position of the first segment in the segment list is determined by `@startNumber`. The segment list then provides the explicit URLs. (NOTE: This is not properly documented in ISO/IEC 23009-1 and requires a correction).

6.4.6.3 Number-based template

If the Representation contains or inherits a **`SegmentTemplate`** element with *$Number$* then the URL of the media segment at position *k* is obtained by replacing the $*Number*$ identifier by (*k*-1) + `@startNumber` in the **`SegmentTemplate`**`@media` string.

6.4.6.4 Time-based template

If the Representation contains or inherits a **`SegmentTemplate`** element with *$Time$* then the URL of the media segment at position *k* is obtained by replacing the $*Time*$ identifier by *MST*(*k*) (de-normalized with the value of the `@timescale` attribute) in the **`SegmentTemplate`**`@media` string.

6.4.7 Scheduling playout

The client schedules the playout based on the available information in the MPD.

The media presentation time in a Period is determined for each Representation as presentation time value in the media segments minus the value of the `@presentationTimeOffset`, if present, for each Representation.

Each segment at position *k* has assigned an earliest media presentation time *EPT*(*k*).

By offering an MPD it is guaranteed that

1. each segment in this Period is available prior to its earliest presentation time and its duration, i.e., for all *k*,

$$SAST(k) <= EPT(k) + (AST + PS) + MD(k)$$

2. If each segment with segment number *k* is delivered starting at *SAST*(*k*) over a constant bitrate channel with bitrate equal to value of the `@bandwidth` attribute then each presentation time *PT* is available at the client latest at time *PT* + (*AST* + *PS*) + *MBT* + *MD*(*k*)
3. A recommended playout-time *MPTS* (*PT*) for a presentation time when operating in sync with other clients is *MPTS*(*PT*) = (*AST* + *PS*) + *PT* + *SPD.*
4. Each segment in this Period is available at least until *SAST*(*k*) + *TSB* + *MD*(*k*).

Using this information, the client can now start scheduling playout taking into account the information in the MPD as well the download speed.

A suitable playout time is *POT*(*PT*) = *MPTS*(*PT*), if the attribute `@suggestedPresentationDelay` is present. If not, then a suitable playout time takes into account the first, second and fourth constraints, i.e., the segment availability times at the server as well as the bitrate variation of the media stream.

6.4.8 Validity of MPD

The MPD can be used to construct and request segments until media time *FT* + *MUP*. 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*. Note that the latest segment may be shorter in duration than the other ones.

6.5 MPD retrieval

It is recommended retrieve content of MPD files using secure HTTP (HTTPS) access. This prevents possible alterations of such files and all sorts of man-in-the middle attacks. Secure retrieval of MPD files is also essential for implementing segment authentication, as described in ISO/IEC 23009-4.

In cases when HTTP server supports gzip and vcdiff compression tools, the effectiveness of downloading and sequential updates of MPD files can be substantially improved. The IETF RFC 3229 provides such a framework. **Figure 27** shows an example of an HTTP request/response for transmitting update of an MPD using vcdiff/gzip encoding:

```
GET /foo.mpd HTTP/1.1
Host: example.com
If-none-match: "abc"
Accept-encoding: gzip
A-IM: vcdiff, gzip


HTTP/1.1 226 IM Used
Date: Wed, 01 Apr 2013 14:01:00 GMT
Delta-base: "abc"
Etag: "ghi"
IM: vcdiff, gzip
```

Figure 27 – Example of using vcdiff and gzip tools for compressed retrieval of MPD files.

6.6 Segment list generation

6.6.1 General

Assume that the DASH client has access to an MPD. This clause describes how a client may generate a Segment list for one Representation as shown in Table 2 from an MPD obtained at *FetchTime* at a specific client-local time *NOW*. In this description, the term *NOW* is used to refer to "the current value of the clock at the reference client when performing the construction of an MPD Instance from an MPD". A client that is not synchronized with a DASH server, which is in turn is expected to be synchronized to UTC, may experience issues in accessing Segments as the Segment availability times provided by the server and the local time *NOW* may not be synchronized. Therefore, DASH clients are expected to synchronize their clocks to a globally accurate time standard.

Table 2 — Segment list in example client

Parameter Name	Cardinality	Description
Segments	1	Provides the Segment URL list.
InitializationSegment	0, 1	Describes the Initialization Segment. If not present each Media Segment is self-initializing.
URL	1	The URL where to access the Initialization Segment (the client may add a byte range to the URL request if one is provided in the MPD).
MediaSegment	1 … N	Describes the accessible Media Segments.
startTime	1	The MPD start time of the Media Segment in the Period relative to the start time of Period.
Duration	1	The MPD duration for the Segment
URL	1	The URL where to access the Media Segment, possibly combined with a byte range.
IndexSegment	1 … N	Describes the accessible Index Segments, if present.
URL	1	The URL where to access the Index Segment, possibly combined with a byte range.

According to ISO/IEC 23009-1 Clause 5.3.9 there exists three different ways to describe and generate a Segment List. This description focuses on the first two where either a **`SegmentList`** element or a **`SegmentTemplate`** element is present. The case with a single Media Segment using **`BaseURL`** element and **`SegmentBase`** element is considered straightforward.

a) If the Representation contains or inherits a **`SegmentTemplate`** element, then the procedures in Clause 6.6.2 are used to generate a list of Media Segments.

b) If the Representation contains or inherits one or more **SegmentList** elements, providing a set of explicit URL(s) for Media Segments, then the procedures in Clause 6.6.3 are used to generate a list of Media Segments.

c) If the **MPD**@type attribute is 'dynamic', then the restrictions on Media Segment Lists as provided in Clause 6.6.4 need to be taken into account.

The client should only request Segments that are included in the Segment list when generated at the actual wall-clock time *NOW*.

6.6.2 Template-based generation of segment list

If the Representation contains or inherits a **SegmentTemplate** element, then the procedures in this subclause are used to generate a list of Segment parameters, i.e. Segment URLs and Media Segment start times..

Assume that the Period end time documented in the current MPD with fetch time *FetchTime* is defined as *PeriodEndTime*. For any Period in the MPD except for the last one, the *PeriodEndTime* is obtained as the value of the *PeriodStart* time of the next Period. For the last Period in the MPD

— if the **MPD**@mediaPresentationDuration attribute is present, then *PeriodEndTime* is defined as the end time of the Media Presentation.

— if the **MPD**@mediaPresentationDuration attribute is not present, then *PeriodEndTime* is defined as *FetchTime* + **MPD**@minimumUpdatePeriod.

For the **SegmentTemplate** element, the relevant identifiers are replaced in the **SegmentTemplate**@media.

Assume that Media Segments within a Representation have been assigned consecutive numbers *i*=@startNumber, @startNumber + 1 i.e. the first Media Segment has been assigned the number *i*=@startNumber, the second Media Segment has been assigned the index *i*=@startNumber+2, and so on. If Index Segments are provided, each Index Segment has an identical index *i* assigned to it.

A valid list of Media Segments with Segment indices *i*, MediaSegment.StartTime[*i*] and MediaSegment.URL[*i*], and if present, a corresponding list of Index Segments with IndexSegment.URL[*i*], *i*=@startNumber, @startNumber + 1, …, is obtained as follows using the @duration attribute for this Representation:

1) Set *i*=@startNumber.

2) The MPD start time of the first Media Segment is 0, i.e. MediaSegment.StartTime[i] = 0.

3) The URL of the Media Segment *i*, MediaSegment.URL[*i*], is obtained by replacing the $*Number*$ identifier by *i* in the template.

4) If Index Segments are present, the URL of the Index Segment *i*, IndexSegment.URL[*i*], is obtained by replacing the $*Number*$ identifier by *i* in the template. Furthermore, any relative URLs are resolved by reference resolution.

5) If ((*PeriodStart* + MediaSegment.StartTime[*i*] + @duration) <= *PeriodEndTime*) then increment *i*, set MediaSegment.StartTime[*i*] = MediaSegment.StartTime[*i*-1] + @duration, and proceed with step 3. Otherwise, continue with step 6.

6) A new Media Segment is added to the list, i.e. *i* = *i* + 1, MediaSegment.StartTime[*i*] = MediaSegment.StartTime[*i*-1] + @duration and the guaranteed duration is set to MediaSegment.duration[*i*] = *PeriodEndTime* - MediaSegment.StartTime[*i*] The restrictions as

specified in A.3.4 are applied for the creation of the accessible list of Media Segments and this concludes Segment List generation.

If instead of the `@duration` attribute a **`SegmentTimeline`** element is given, then the variable durations of the Segments are used to compute the start times and durations. Depending on the identifier, *$Number$* or *$Time$,* the appropriate replacements are done as specified in ISO/IEC 23009-1 Clause **Error! Reference source not found.**. If neither the `@duration` nor the **`SegmentTimeline`** element is given, then the *MediaSegment.StartTime*[*1*] of the only provided Segment is set to 0.

6.6.3 Playlist-based generation of segment list

If the Representation contains or inherits one or more **`SegmentList`** elements, each containing **`SegmentURL`** elements, then the procedures specified in this subclause apply to generate a valid list of accessible Segment URLs and Media Segment start times.

Assume that Media Segments within a Representation have been assigned consecutive indices *i*=`@startNumber`, `@startNumber`+1, …., i.e. the first Media Segment has been assigned *i*=`@startNumber`, the second Media Segment has been assigned *i*=`@startNumber`+1, and so on. If an Index Segment is provided for each Media Segment, each Index Segment has an identical index *i* assigned to it.

If a **`SegmentTimeline`** element has been given, a list of Segment start times and durations is first generated by expanding the **`SegmentTimeline`** from its run-length compressed form into a SegmentTimelineList of StartTime values for each Segment. This new list is indexed starting at `@startNumber`.

A valid list of Media Segments with Segment indices *i*=`@startNumber`, `@startNumber`+1, …, MediaSegment.StartTime[*i*] and MediaSegment.URL[*i*], and if present, a corresponding list of Index Segments with IndexSegment.URL[*i*] is obtained as follows:

1) Set *i*=`@startNumber`.

2) The MPD start time of the first Media Segment is 0, i.e. MediaSegment.StartTime[*i*] = 0.

3) The URL of the Media Segment *i*, *MediaSegment.URL[i]*, is obtained as the **`SegmentURL`**`@media` attribute of the (*i*-`@startNumber` +1)th **`SegmentURL`** element in the **`SegmentList`** element taking into account URI reference resolution, possibly using the byte range specified in the `@mediarange` attribute of the same **`SegmentURL`** element, if present.

4) The URL of the Index Segment *i*, *IndexSegment.URL[i]*, is obtained also from the **`SegmentURL`** element or inherited from above

5) If the `@duration` attribute is provided, then the *MediaSegment.StartTime[i]* of Media Segment *i* is obtained as (*i*-`@startNumber` -1)*`@duration`. If the `@duration` attribute is not provided and a **`SegmentTimeline`** element is in effect then the variable durations are taken into account for the computation of the start times. Otherwise, the *MediaSegment.StartTime[1]* of the only provided Segment is set to 0.

6) If this is not the last **`SegmentURL`** element, a new Media Segment is added to the list, i.e. *i* = *i* + 1, and proceed with step 2; Otherwise continue with step 5.

7) The restrictions as specified in Clause 6.6.4 are applied for the creation of the accessible list of Media Segments. This concludes Segment list generation.

6.6.4 Media segment list restrictions

The Media Segment List is restricted to a list of accessible Media Segments, which may be a subset of the Media Segments of the complete Media Presentation. The construction is governed by the current value of the clock at the client *NOW*.

Generally, Segments are only available for any time *NOW* between **MPD**@availabilityStartTime and **MPD**@availabilityEndTime. For times *NOW* outside this window, no Segments are available.

In addition, for services with **MPD**@type='dynamic', assume the variable *CheckTime* associated to an MPD with *FetchTime* is defined as:

1) If the **MPD**@minimumUpdatePeriod attribute in the client is provided, then the check time is defined as the sum of the fetch time of this operating MPD and the value of this attribute, i.e. *CheckTime* = *FetchTime* + **MPD**@minimumUpdatePeriod.

2) If the **MPD**@minimumUpdatePeriod attribute in the client is not provided, external means are used to determine *CheckTime*, such as a priori knowledge, or HTTP cache headers, etc.

The *CheckTime* is defined on the MPD-documented media time axis; when the client's playback time reaches *CheckTime* - **MPD**@minBufferTime it should fetch a new MPD.

Then, the Media Segment list is further restricted by the *CheckTime* together with the MPD attribute **MPD**@timeShiftBufferDepth such that only Media Segments for which the sum of the start time of the Media Segment and the Period start time falls in the interval [*NOW*- **MPD**@timeShiftBufferDepth - @duration, min(*CheckTime, NOW*)] are included.

6.7 Rate adaptation

When designing rate adaptation algorithm for DASH, it is important to recognize that:

- @bandwidth attributes may not provide accurate information about rate at which each segment is encoded; therefore rate estimation should be based on information in:
    - segment index files, and/or
    - actual length values returned by processing of HTTP GET requests.

It is also important to take into account the following considerations:

- that the rate adaptation algorithm is efficiently utilizing the sharable network capacities, which affects playback media quality,
- that the rate adaptation algorithm is capable of detecting network congestion and is able to react promptly to prevent playback interruption,
- that the rate adaptation algorithm can provide stable playback quality even if the network delivery capacities fluctuate widely and frequently,
- that the rate adaptation algorithm is able to tradeoff maximum instantaneous quality and smooth continuous quality, for example by smoothing short-term fluctuation in the network delivery capacities by using buffering, but still switch to better presentation quality/higher bitrates if more long-term bandwidth increase is observed,
- that the rate adaptation algorithm is able to avoid excessive bandwidth consumption due to over-buffering media data.

When implementing rate adaptation in DASH, one could balance between different criteria listed above to improve the overall Quality of Experience (QoE) perceived by the user.

In absence of other information, e.g. from the radio network status, the measurement for certain QoE metrics may be used in rate adaptation in DASH, e.g.:

- average throughput: average throughput measured by a client in a certain measurement interval;
- Segment Fetch Time (SFT) ratio: the ratio of Media Segment Duration (MSD) divided by SFT. MSD and SFT denote the media playback time contained the media segment and the period of time from the time

instant of sending a HTTP GET request for the media segment to the instant of receiving the last bit of the requested media segment, respectively;

- buffer level: buffered media time at a client.

6.8 Seeking

Assume that a client attempts to seek to a specific Media Presentation time T_M in a Representation relative to the *PeriodStart* time. According to ISO/IEC 23009-1 Clause **Error! Reference source not found.**, the presentation times within each Period are relative to the *PeriodStart* time of the Period minus the value of the `@presentationTimeOffset`, T_O, of the containing Representation.

Based on the MPD, the client has access to the MPD start time and Media Segment URL of each Segment in the Representation, along with Index Segment URL, if present. The Segment number of the Segment most likely to contain media samples for Media Presentation time T_M is obtained as the maximum Segment index *i**, for which the start time MediaSegment[*i*].StartTime is smaller or equal to the T_M. The Segment URL is obtained as MediaSegment[*i**].URL.

Note that timing information in the MPD may be approximate due to issues related to placement of Stream Access Points, alignment of media tracks and media timing drift. As a result, the Segment identified by the procedure above may begin at a time slightly after T_M and the media data for presentation time may be in the previous Media Segment. In case of seeking, either the seek time may be updated to equal the first sample time of the retrieved Media Segment, or the preceding Media Segment may be retrieved instead. However, note that during continuous playout, including cases where there is a switch between alternative versions, the media data for the time between T_M and the start of the retrieved Segment is always available.

For accurate seeking to a presentation time T_M, the DASH Client needs to access Stream Access Points (SAP). To determine the SAP in a Media Segment in case of DASH, the client may, for example, use the information in the Segment Index if present to locate the random access points and the corresponding presentation time in the Media Presentation.

In the case that the Media Presentation is based on the ISO base media file format and a Segment is a movie fragment, it is also possible for the client to use information within the '`moof`' and '`mdat`' boxes, for example, to locate Stream Access Points in the media and obtain the necessary presentation time from the information in the movie fragment and the Segment start time derived from the MPD. If no SAP with presentation time before the requested presentation time T_M is available, the client may either access the previous Segment or may just use the first SAP as the seek result. When Media Segments start with a SAP, these procedures are simplified.

In the case that the Media Presentation is based on MPEG-2 TS, the presentation units corresponding to the desired presentation time T_M can be identified by using the indexing information, if present, in conjunction with the differential value of the presentation time stamps (PTS) within the Media Segment. For example, if $T_{M,S}$ denotes the presentation time corresponding to the last SAP leading the desired seek time *tp*, with a corresponding PTS denoted as PTS^s, then the desired seek position within the media has a PTS expressed as: $((T_M - T_{M,S}) * \text{timescale} + PTS_S \% 2^{33}$.

Also note that not necessarily all information of the Media Segment needs to be downloaded to access the presentation time T_M. The client may for example initially request the Segment Index from the beginning of the Media Segment using partial HTTP GET. By use of the Segment Index, Segment timing can be mapped to byte ranges of the Segment. By continuously using partial HTTP GET requests, only the relevant parts of the Media Segment may be accessed for improved user experience and low start-up delays.

6.9 Support for trick modes

The client may receive user command to pause or stop a Media Presentation. In this case client may instruct media engine(s) to pause or stop playback, and then stop requesting new Media Segments or parts thereof. To resume, the client may send requests to Media Segments, starting with the next Subsegment after the last requested Subsegment.

If a specific **Representation** or **SubRepresentation** element includes the @maxPlayoutRate attribute, then the corresponding Representation or Sub-Representation may be used for the fast-forward trick mode. The client may play the Representation or Sub-Representation with any speed up to the regular speed times the specified @maxPlayoutRate attribute with the same decoder profile and level requirements as the normal playout rate. If a specific **Representation** or **SubRepresentation** element includes the @codingDependency attribute with value set to 'false', then the corresponding Representation or Sub-Representation may be used for both fast-forward and fast-rewind trick modes.

Sub-Representations in combination with Index Segments and Subsegment Index boxes may be used for efficient trick mode implementation. Given a Sub-Representation with the desired @maxPlayoutRate, ranges corresponding to **SubRepresentation**@level all level values from **SubRepresentation**@dependencyLevel may be extracted via byte ranges constructed from the information in Subsegment Index Box. These ranges can be used to construct more compact HTTP GET request.

6.10 Stream switching

Based on information during an ongoing Media Presentation, a client may decide to switch Representations. Switching to a “new” Representation is equivalent to tuning in or seeking to the new Representation from the time point where the "old" Representation has been presented. Once switching is desired, the client should seek to a SAP for each media stream in the “new” Representation at a desired presentation time t_p later than and close to the current presentation time. Presenting the “old” Representation up to the SAP in the “new” Representation that enables seamless switching.

If @segmentAligment is set true and the @startWithSAP is set to 1, 2 or 3 (and in the latter case the **Representation**@mediaStreamStructureId is identical for the two Representations), then the client may switch at any Segment boundary by just concatenating Segments with consecutive indices from different Representations. No overlap downloading and decoding is required.

The same can be achieved on Subsegment level with @SubsegmentAlignment set true and @SubsegmentStartsWithSAP the same values and conditions as above.

In cases when client implements switching between Representations that:

- have significant gap in bitrate;
- have different resolutions or sampling rates,
- use different codecs/profiles or audio types;

or other factors that may introduce discontinuities or have diminishing effect on user experience, the client may use signal processing techniques to smooth such transitions. For example, this can be done by downloading overlapping segments, decoding audio or video content and then cross-fading the results prior to the playback.

6.11 Client support for dependent representations

6.11.1 General

In the ISO base Media File Format on Demand profile, the DASH Client performs adaptations based on Subsegments. For this purpose, the Clients should download the Segment Indexes for the Representations and parse them to get the byte-ranges at which the client can access the Subsegments of the offered Representations. Once the Client has this information it can switch from one Representations to another at any Subsegment. In the following two examples of the Subsegments received at a client with switching events are shown.

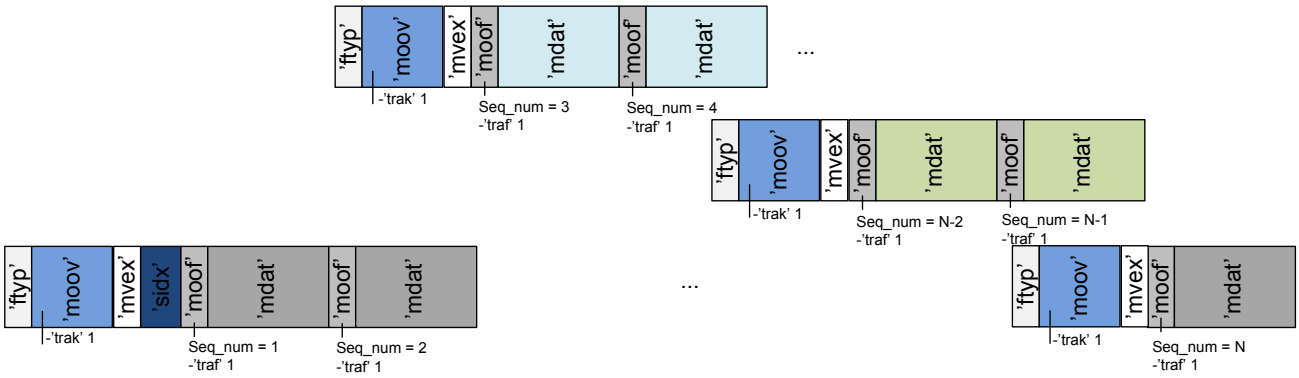

Figure 28 – Example of Subsegment sequence received for non-dependent Representations.

For the case shown in Figure 28, where all Representation in an Adaptation Set are independent from each other, i.e. there are not dependent Representations, the ISO base Media File Format on Demand profile allows clients to switch seamlessly at any Subsegment. However, since the 'moov' box contained within the Segments of the different Representations is typically different whenever a client switches from one to another Representation the 'moov' has to be updated and the content is treated as a new file.

In Figure 29, an example of a Subsegment sequence received at the DASH Clients for dependent Representations is shown. The main difference to the example shown above is mainly that the Subsegments are combined in a single file, which is the same when switching events occur.

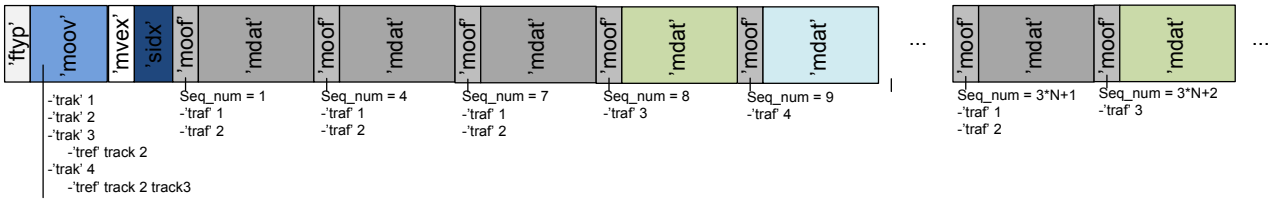

Figure 29 – Example of Subsegment sequence received for dependent Representations.

As mentioned before, the 'moov' is common to all Representation, therefore switching from one Representation to another implies missing or adding movie fragments of another track. It requires at the DASH Client side to play a different track depending on the number of representations received (e.g. SVC with extractors in higher layers) or omitting the playback of a track (e.g. MVC without extractors when a view is missing). For instance for the case when the 3 Representations shown in the example contain SVC layers with their corresponding extractors, the for the first two movie fragments shown in the figure track 1 and track 2 should be played, for the following movie fragments track 1 and track 4 should be played and for the last shown movie fragment track 1 and track 3 should be played.

6.11.2 Client trick-mode support using SubRepresentations

For the case where trick modes or bitstream thinning are performed based on accessing sub-representations of the original data, DASH Clients should take special care to not to try to access non downloaded data. Since the DASH Client is aware of the level it has downloaded, with information on the 'leva' box, track_IDs, and 'strk' box the Client has sufficient information to prevent playing non downloaded data.

Special care has to be taken when the assignment type is other than 3, since the 'mdat' boxes show a different length than the downloaded amount of data. If the padding_flag in the 'leva' box is set, then the non-downloaded data can be filled by zeros. If not the length of the 'mdat' box should be changed. If the assignment type is equal to 3 all downloaded 'mdat' contain the whole data for their length.

If this data is stored for usage in the future, then the player will not be aware of the downloaded data. Therefore, the downloaded data when Sub-Representations are used should be stored as incomplete Tracks.

6.12 Events

TBA

7 Extending DASH

7.1 Extension of MPD Schema in external namespace

7.1.1 General

In order to extend the DASH namespace in an external namespace, and in particular avoid failing XML schema rule Unique Particle Attribution (ref . http://www.w3.org/TR/xmlschema-1/#cos-nonambig). the following is proposed:

1. Remove the wildcard instruction <xs:any namespace="##other" … /> just after the optional element from another namespace
2. Specify client rules that would make the client ignore and remove all unknown elements and attributes from *another* namespace prior to attempt any validation of an MPD instantiation.
3. Specify client rules that would make the client ignore and remove all unknown elements and attributes from the *target* namespace prior to attempt any validation of an MPD instantiation.

An example is provided in 7.1.2 below.

7.1.2 Example

```
<?xml version="1.0"?>
<xs:schema targetNamespace="urn:mpeg:dash:schema:mpd:2011"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xExtension="urn:example:com:dash-extension"
  xmlns="urn:mpeg:dash:schema:mpd:2011">

  <xs:import namespace="http://www.w3.org/1999/xlink" schemaLocation="xlink.xsd"/>
  <xs:import namespace=" urn:example:com:dash-extension" schemaLocation="dash-ext.xsd"/>

  <xs:annotation>
    <xs:appinfo>Media Presentation Description</xs:appinfo>
    <xs:documentation xml:lang="en">
      This Schema defines the Media Presentation Description.
    </xs:documentation>
  </xs:annotation>

  <!-- MPD: main element -->
  <xs:element name="MPD" type="MPDtype"/>

  <!-- MPD Type -->
  <xs:complexType name="MPDtype">
    <xs:sequence>
      <xs:element name="ProgramInformation" type="ProgramInformationType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Location" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Period" type="PeriodType" maxOccurs="unbounded"/>
      <xs:element name="Metrics" type="MetricsType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="xExtension:ExtendedElement" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
...
  </xs:complexType>

....
```

8 Bibliography

[SW11] V. Swaminathan and Sheng Wei, "Low latency live video streaming using HTTP chunked encoding", Proc. 2011 IEEE 13th International Workshop on Multimedia Signal Processing (MMSP), Hangzhou, China, October 2011.