ETSI TS 126 247 V10.4.0 (2013-07)

Universal Mobile Telecommunications System (UMTS);
LTE;
Transparent end-to-end
Packet-switched Streaming Service (PSS);
Progressive Download and Dynamic
Adaptive Streaming over HTTP (3GP-DASH)
(3GPP TS 26.247 version 10.4.0 Release 10)

Reference
RTS/TSGS-0426247va40

Keywords
LTE,UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

Contents

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

The 3GPP transparent end-to-end packet-switched streaming service (PSS) specification consists of seven 3GPP TSs: 3GPP TS 22.233 [1], 3GPP TS 26.233 [2], 3GPP TS 26.234 [3], 3GPP TS 26.244 [4], 3GPP TS 26.245 [5], 3GPP TS 26.246 [6], and the present document.

The TS 22.233 contains the service requirements for the PSS. The TS 26.233 provides an overview of the PSS. The TS 26.234 provides the details of the protocols and codecs used by the PSS. The TS 26.244 defines the 3GPP file format (3GP) used by the PSS and MMS services. The TS 26.245 defines the Timed text format used by the PSS and MMS services. The TS 26.246 defines the 3GPP SMIL language profile. The present document defines Progressive Download and Dynamic Adaptive Streaming over HTTP.

The TS 26.244, TS 26.245 and TS 26.246 start with Release 6. Earlier releases of the 3GPP file format, the Timed text format and the 3GPP SMIL language profile can be found in TS 26.234.

The TS 26.247 starts with Release 10. Earlier releases of Progressive Download and Dynamic Adaptive Streaming over HTTP can be found in TS 26.234.

Introduction

Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH) collects a set of technologies how progressive download and adaptive streaming of continuous media may be carried out exclusively over HTTP.

1 Scope

The present document specifies Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH). This specification is part of Packet-switched Streaming Service (PSS). HTTP-based progressive download and dynamic adaptive streaming are separated from TS 26.234 to differentiate from RTP-based streaming that is maintained in TS 26.234. HTTP-based progressive download and dynamic adaptive streaming may be deployed independently from RTP-based PSS, for example by using standard HTTP/1.1 servers for hosting data formatted as defined in the present document.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TS 22.233: "Transparent End-to-End Packet-switched Streaming Service; Stage 1".

[2] 3GPP TS 26.233: "Transparent end-to-end Packet-switched Streaming service (PSS); General description".

[3] 3GPP TS 26.234: "Transparent end-to-end packet switched streaming service (PSS); Protocols and codecs".

[4] 3GPP TS 26.244: "Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP)".

[5] 3GPP TS 26.245: "Transparent end-to-end packet switched streaming service (PSS); Timed text format".

[6] 3GPP TS 26.246: "Transparent end-to-end packet switched streaming service (PSS); 3GPP SMIL Language Profile".

[7] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[8] IETF STD 0007: "Transmission Control Protocol", Postel J., September 1981.

[9] IETF RFC 2616: "Hypertext Transfer Protocol – HTTP/1.1", Fielding R. et al., June 1999.

[10] Open Mobile Alliance, Service and Content Protection for Mobile Broadcast Services, Approved Version 1.0, February 2009.

[11] ISO/IEC 14496-12: | 15444-12:: "Information technology – Coding of audio-visual objects – Part 12: ISO base media file format" | "Information technology – JPEG 2000 image coding system – Part 12: ISO base media file format".

[12] IETF RFC 2818: "HTTP Over TLS", E. Rescorla, May 2000.

[13] IETF RFC 5646: "Tags for Identifying Languages", A. Phillips, M. Davis, September 2009.

[14] (void)

[15] Open Mobile Alliance: "DRM Content Format V 2.0".

[16] Open Mobile Alliance: "DRM Content Format V 2.1".

[17] IETF RFC 3986: "Uniform Resource Identifiers (URI): Generic Syntax", Berners-Lee T., Fielding R. and Masinter L., January 2005.

[18] IETF RFC 1952: "GZIP file format specification" version 4.3, P. Deutsch, May 1996.

[19] IETF RFC 1738: "Uniform Resource Locators (URL)", December 1994.

[20] W3C XLINK: "XML Linking Language (XLink)" Version 1.1, W3C Recommendation 06, May 2010.

[21] IETF RFC 3406: "Uniform Resource Names (URN) Namespace Definition Mechanisms", October 2002.

[22] OMA-ERELD-DM-V1_2-20070209-A: "Enabler Release Definition for OMA Device +Management, Approved Version 1.2"

[23] 3GPP TS 33.310: "Network Domain Security (NDS); Authentication Framework (AF)".

[24] IETF RFC 2045: "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies".

[25] IETF RFC 2231: " MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations".

[26] IETF RFC 6381: " The 'Codecs' and 'Profiles' Parameters for "Bucket" Media Types," August 2011.

[27] Void.

[28] IEEE 1003.1-2008, IEEE Standard for Information Technology - Portable Operating System Interface (POSIX), Base Specifications, Issue 7

[29] IETF RFC 4337, "MIME Type Registration for MPEG-4," March 2006

[30] IETF RFC 3023, "XML Media Types," January 2001.

[31] 3GPP TS 23.203: "Policy and charging control architecture".

[32] 3GPP TS 29.213: "Policy and Charging Control signalling flows and Quality of Service (QoS) parameter mapping".

[33] 3GPP TS 29.214: "Policy and Charging Control over Rx reference point".

[34] IETF RFC 3629: "UTF-8, a transformation format of ISO 10646," November 2003.

[35] IETF RFC 4288: "Media Type Specifications and Registration Procedures," December 2005.

[36] IETF RFC 4648: "The Base16, Base32, and Base64 Data Encodings," October 2006.

[37] ISO/IEC 23009-1: "Information technology – Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats".

3 Definitions, abbreviations and conventions

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TR 21.905 [7] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in TR 21.905 [7].

access unit: unit of a media stream with an assigned Media Presentation time.

accessibility: the degree to which a media content or certain media content components are available to as many people as possible.

Adaptation Set: a set of interchangeable encoded versions of one or several media content components.

availableSegment: Segment which is accessible at its assigned HTTP-URL, possibly restricted by a byte range, i.e. the request with an HTTP GET results in a reply of the Segment and a 2xx OK status code.

continuous media: media with an inherent notion of time. In the present document speech, audio, video, timed text and timed graphics.

DASH metric: a metric identified by key and defined in this part of the specification.

earliest presentation time: the smallest presentation time of any access unit of a Media Segment or Subsegment for a media stream.

group: collection of Representations that are expected to not being presented jointly.

HTTP-URL: a URI with a fixed scheme of 'http' or https.

Initialization Segment: Segment containing metadata that is necessary to present the media streams encapsulated in Media Segments.

media content: one media content period or a contiguous sequence of media content periods.

media content component: one continuous component of the media content with an assigned media component type that can be encoded individually into a media stream.

media content component type: a single type of media content such as audio, video, or text.

media content period: set of media content components that have a common timeline as well as relationships on how they may be presented.

Media Presentation: collection of data that establishes a bounded or unbounded presentation of media content.

Media Presentation Description (MPD): formalized description for a Media Presentation for the purpose of providing a streaming service.

Media Presentation timeline: concatenation of the timeline of all Periods which itself is common to all Representations in the Period.

Media Segment: Segment that complies with media format in use and enables playback when combined with zero or more preceding Segments, and an Initialization Segment (if any).

media stream: encoded version of a media content component.

Media Subsegment: Subsegment that only contains media data but no Segment Index.

MPD start time: approximate presentation start time of a Media Segment signalled in MPD.

MPD duration: approximate presentation duration of a Media Segment signalled in MPD.

Period: interval of the Media Presentation, where a contiguous sequence of all Periods constitutes the Media Presentation.

presentation time: a time associated to an access unit that maps it to the Media Presentation timeline.

remote element: one or more elements that are not fullly contained in the MPD document but is referenced in the MPD with an HTTP-URL.

Representation: collection and encapsulation of one or more media streams in a delivery format and associated with descriptive metadata.

Segment: smallest addressable unit in an MPD with a defined format.

Segment availability end time: the time instant in wall-clock time at which a Segment ceases to be an available Segment.

Segment availability start time: the time instant in wall-clock time at which a Segment becomes an available Segment.

Segment Index: a compact index of the time range to byte range mapping within a Media Segment separately from the MPD.

Stream Access Point (SAP): 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).

Sub-Representation: part of a Representation described in the MPD that is present in the entire Period.

Subsegment: smallest unit within Media Segments that is indexed by a Segment Index.

valid Segment URL: an HTTP-URL that is promised to reference a Segment during its Segment availability period.

wall-clock time: time as stated by UTC (Universal Co-ordinated Time).

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [7] and the following apply.
An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in TR 21.905 [7].

3GP	3GPP file format
3GP-DASH	3GPP Dynamic Adaptive Streaming over HTTP
AHS	Adaptive HTTP Streaming
AVC	Advanced Video Coding
DM	Device Management
DRM	Digital Rights Management
HSD	HTTP Streaming and Download
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDR	Instantaneous Decoding Refresh
MPD	Media Presentation Description
MPEG-2 TS	Moving Picture Experts Group Transport Stream
MIME	Multipurpose Internet Mail Extensions
OMA	Open Mobile Alliance
PDCF	Packetized DRM Content Format
PSS	Packet-switched Streaming Service
QoE	Quality-of-Experience
RFC	Request For Comments
RTP	Real-time Transport Protocol
SAP	Stream Access Point
SMIL	Synchronised Multimedia Integration Language
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UTC	Universal Time Coordinated
UTF-8	Unicode Transformation Format (the 8-bit form)
UUID	Universally Unique Identifier
W3C	WWW Consortium
XML	eXtensible Markup Language
XSLT	eXtensible Stylesheet Language Transformation

3.3 Conventions

The following naming conventions apply in this specification:

- Elements in an XML-document are identified by an upper-case first letter and in bold face as **`Element`**.
  To express that an element **`Element1`** is contained in another element **`Element2`**, we may write **`Element2.Element1`**. If an element is constructed of two or more combined words, camel-casing is typically

used, e.g. **`ImportantElement`**. Elements are present exactly once, or the minimum and maximum occurence is defined by <minOccurs> ... <maxOccurs>.

- Attributes in an XML-document are identified by a lower-case first letter as well as they are preceded by a "@"-sign, e.g. `@attribute`. To point to a specific attribute `@attribute` contained in an element **`Element`**, we may write **`Element`**`@attribute`. If an attribute is constructed of two or more combined words, camel-casing is typically used after the first word, e.g. `@veryImportantAttribute`. Attributes are assigned a status in the XML as mandatory (M), optional (O), optional with default value (OD) and conditionally mandatory (CM).

- Namespace qualification of elements and attributes is used as per XML standards, in the form of **`namespace:Element`** or `@namespace:attribute` The fully qualified namespace will be provided in the schema fragment associated with the declaration. This specification extends the namespace of DASH, by documenting the element name in the semantic table with an extension namespace prefix.

- Variables defined in the context of the present document are specifically highlighted with *italics*, e.g. *InternalVariable*.

- Structures that are defined as part of the hierarchical data model are identified by an upper-case first letter, e.g. Media Presentation, Period, Group, Adaptation Set, Representation, Segment, etc.

4 Overview

The present document specifies Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH) for continuous media. The features are separated from the umbrella specification TS 26.234 [3] to differentiate from RTP-based streaming that is specified and maintained in TS 26.234. Services relying exclusively on these features may be deployed independently from RTP-based PSS servers, for example by using standard HTTP/1.1 servers for hosting the services.

The specification covers the following aspects:

- System Description: describes the relationship to the PSS architecture and refines the architecture, interfaces and protocols that are defined in this specification.

- Progressive Download over HTTP.

- 3GPP Dynamic Adaptive Streaming over HTTP (3G-DASH) provides an overview of the architecture, the formats and the models that build the basis for 3GP-DASH. Also, 3GP-DASH Profiles provides an identifier and refers to a set of specific restrictions in this or other specifications.

- DASH - Media Presentation describes the data model of a Media Presentation. It also provides an overview on elements and attributes that may be used to describe components and properties of a media presentation in a Media Presentation Description (MPD).

- DASH - Usage of the 3GP file format defines how segments can be formed based on the 3GP file format.

- Quality-of-Experience for Progressive Download and 3GP-DASH.

- Normative annexes for MPD schema (Annex B), Descriptor Scheme Definitions (Annex C), OMA DM QoE Management Object (Annex F), File format extensions for 3GPP DASH support (Annex G) and MIME Type Registration for MPD (Annex H).

- Informative annexes for Client Behaviour (Annex A), MPD Examples (Annex D), and Mapping MPD structure and semantics to SMIL (Annex E).

5 System Description

5.1 Overview

Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH) enables to provide services to deliver continuous media content over Hypertext Transfer Protocol (HTTP) in a sense that all resources that compose the service are accessible through HTTP-URLs and the HTTP/1.1 protocol as specified in RFC 2616 [9] may be used to deliver the metadata and media data composing the service. This enables that standard HTTP servers and standard HTTP caches can be used for hosting and distributing continuous media content. Figure 1 shows the architecture for services using progressive download and Figure 2 shows the architecture for services using 3GP-DASH.

The present document deals with the specification of interfaces between the Client and the Server. Specifically, it defines the formats that may be delivered exclusively over the HTTP interface to enable progressive download and streaming
services.

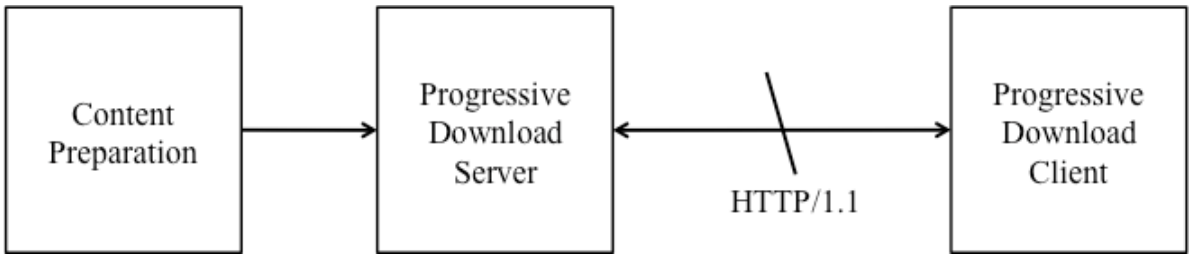

Figure 1: Architecture for Progressive Download over HTTP

Services using the features described in this specification may be deployed within PSS as specified in TS 26.233 [2] and TS 26.234 [3]. In this case the Progressive Download/3GP-DASH Server may be a sub-function of the PSS server and the Progressive Download/3GP-DASH client may be a sub-function of the PSS client.

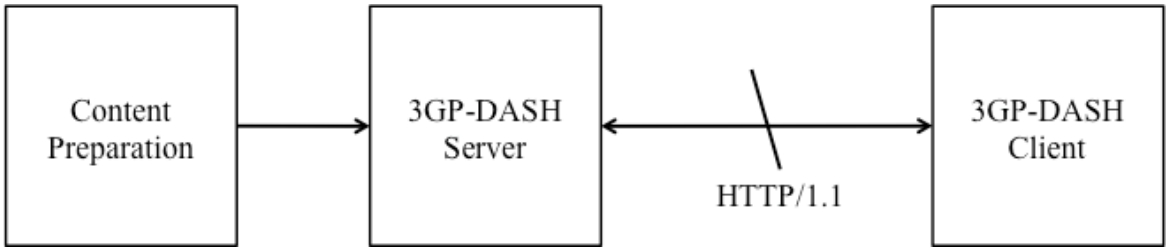

Figure 2: Architecture for 3GP-DASH

Services using the features defined in this specification may also be deployed independent of the PSS servers and clients. In this case the Progressive Download/3GP-DASH client shall support the formats and codecs according to this specification.

Access to services based on the features defined in the present document is introduced in clause 5.2.

The protocol support for services using the features defined in this specification is provided in clause 5.3.

Clients supporting progressive download-based services shall support the features and formats as specified in clause 6 of this specification.

Clients supporting 3GP-DASH shall support the features and formats as specified in clause 7 of this specification.

Clients supporting QoE Metrics and Reporting shall support the features as specified in clause 10 of this specification.

5.2 Service Access

Service access refers to the method by which a Client initially accesses the service. Service access for services based in the specification can be achieved e.g. by a Media Presentation Description or a URL to the media file.

The service access URL can be made available to a client in many different ways. Clients supporting services based on the features in this specification shall be able to access services that are provided through an HTTP-URL. However, it is out of the scope of this specification to mandate any specific mechanism. A preferred way may be to embed URLs for service establishment within HTML pages.

5.3 Protocols

Progressive Download and 3GP-DASH clients shall comply with a *client* as specified in RFC 2616 [9]. The resource hosting the 3GP files and DASH Segments shall comply with a *server* as specified in RFC 2616 [9].

Progressive Download and 3GP-DASH clients should use the HTTP GET method or the HTTP partial GET method, as specified in RFC 2616 [9], clause 9.3, to access media offered at HTTP-URLs.

Figure 3 shows a protocol stack for services in the context of this specification. 3GP Files in progressive download as well as Segments based on the 3GPP File Format shall be accessible through HTTP.

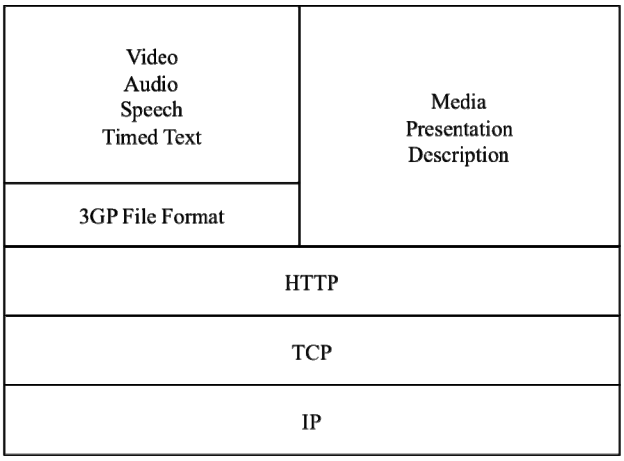

Figure 3: Overview of the protocols stack

Transport security in Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH) is achieved using the HTTPS (Hypertext Transfer Protocol Secure) specified in RFC 2818 [12] and TLS as specified in TLS profile of Annex E in TS 33.310 [23]. In case secure delivery is desired, HTTPS should be used to authenticate the server and to ensure secure transport of the content from server to client.

NOTE: The use of HTTPS for delivering Media Segments may inhibit caching at proxies and add overhead at the server and the client.

6 Progressive Download over HTTP

6.1 General

As an alternative to conventional streaming, a client may download, typically through HTTP, a media file that encapsulates continuous media and may play the media from the local storage. A PSS client shall support progressive download and playout of 3GP files [4] as specified in the remainder of this clause.

The media file encapsulating the continuous media is accessed directly by issuing one or more HTTP GET or partial GET requests to the referenced media file. An example of a valid URL is http://example.com/morning_news.3gp.

6.2 Progressive Download

Progressive download uses normal HTTP download using HTTP GET or partial GET requests. The differences between regular download and Progressive Download are that 1) the content may be authored as progressively downloadable, and 2) the terminal recognises that the content is suitable for progressive download.

A client downloading continuous media may decide to start playout of the encapsulated media data before the download of the media file is completed.

6.3 3GPP File Format Profiles

The following profiles of the 3GPP file format in TS 26.244 [4] shall be supported by clients supporting Progressive Download over HTTP:

- Basic profile, and
- Progressive-download profile.

7 3GPP Dynamic Adaptive Streaming over HTTP

7.1 System Description

The 3GPP Dynamic Adaptive Streaming over HTTP (3GP-DASH) specified in this specification provides streaming services over HTTP. For this it specifies XML and binary formats that enable delivering content from standard HTTP servers to an HTTP-Streaming client and enables caching content by standard HTTP caches.

The specification for 3GP-DASH primarily defines two formats:

1) 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* and to provide the context for these identified resources within a Media Presentation. For 3GP-DASH, the resource identifiers are exclusively HTTP-URLs possibly combined with a byte range.

2) The Segment formats specify the formats of the entity body of the HTTP response to an HTTP GET request or an HTTP partial GET request with the indicated byte range through HTTP/1.1 as defined in RFC 2616 [9] to a resource identified in the MPD. Segments typically contain efficiently coded media data and metadata according to or aligned with common media formats..

The MPD provides sufficient information for a client to provide a streaming service to the user by accessing the Segments through the protocol specified in the scheme of the defined resources, in the context of this specification exclusively HTTP/1.1. Such a client is referred to as a 3GP-DASH client in the remainder of the present document. However, this specification does not provide a normative definition for such a client. An informative client model to illustrate the formats defined in this specification is provided in section 7.2. An informative example client behaviour description is provided in Annex A of this specification.

Figure 7-1 shows an architecture in which the formats defined in this specification are typically used. Boxes with solid lines indicate devices that are mentioned in this specification as they host or process the formats defined in this specification whereas dashed boxes are conceptual or transparent. This specification deals with the definition of formats that are accessible on the interface to the 3GP-DASH client, indicated by the solid lines. Any other formats or interfaces are not in scope of this specification. In the considered deployment scenario, it is assumed that the 3GP-DASH client has access to an MPD. The MPD provides sufficient information for the 3GP-DASH client to provide a streaming service to the user by requesting Segments from an HTTP server and demultiplexing, decoding and rendering the included media streams.

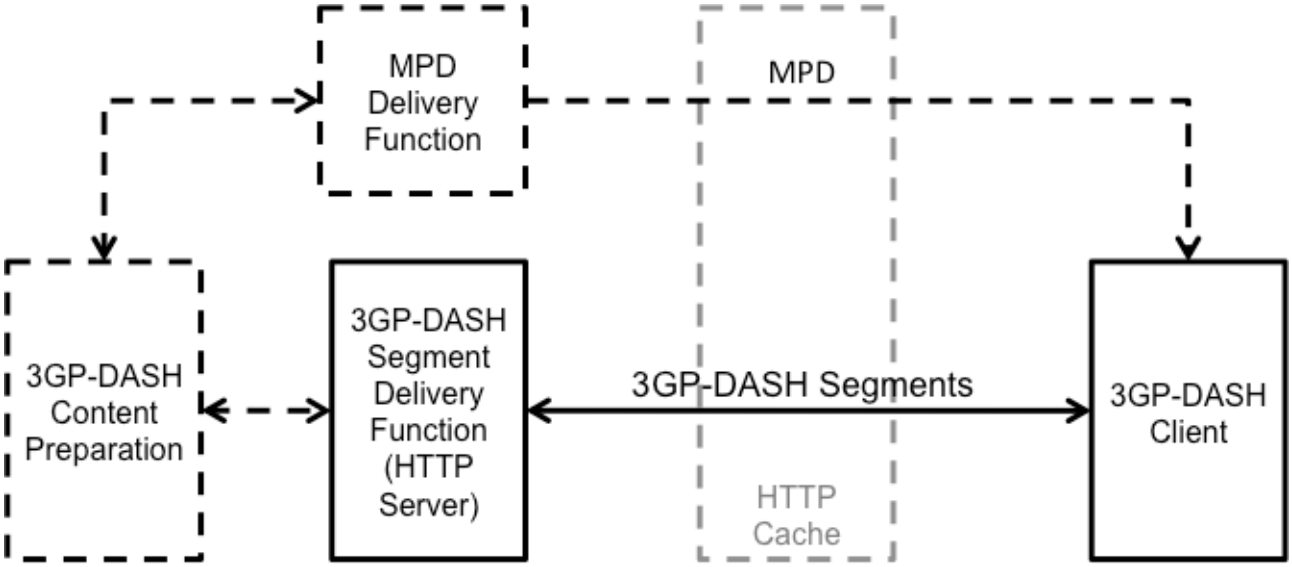

Figure 7-1: System Architecture for 3GP-DASH

The normative aspects of 3GP-DASH formats are defined by

- the profiles defined in clause 7.3.
- the DASH Media Presentation as defined in clause 8.
- the usage of the 3GPP file format for DASH as defined in clause 9.

The clauses mentioned above may refer to normative aspects in clause 10 on Quality-of-Experience as well as to normative Annexes B, C, E, G, and H.

7.2 3GP-DASH Client Model

The design of the formats defined in this specification is based on the informative client model as shown in Figure 7-2. The figure illustrates the logical components of a conceptual 3GP-DASH client model. In this figure the 3GP-DASH Access Engine receives the Media Presentation Description (MPD), constructs and issues requests and receives Segments or parts of Segments. In the context of this standard, the output of the DASH Access Engine consists of media in container formats according to the ISO/IEC 14496-12 ISO Base Media File Format [11] and specifically the 3GP file format [4]. In addition, timing information is provided that maps the internal timing of the media to the time line of the Media Presentation.

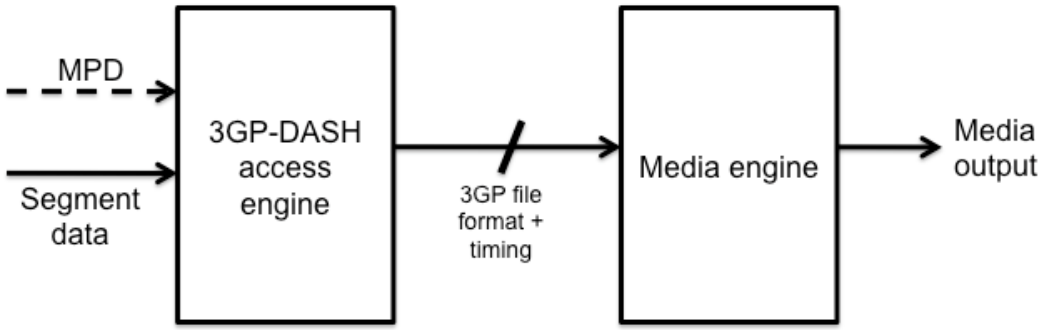

Figure 7-2: 3GP-DASH client Model

7.3 3GP-DASH Profiles

7.3.1 General

Profiles of 3GP-DASH are defined so as to enable interoperability and the signaling of the use of features etc. A profile refers to a set of specific restrictions. Those restrictions might be on features of the MPD as defined in clause 8 of this specification, Segment formats as for example defined in clause 9 of this specification, usage of the network, codec(s) used, content protection formats, or on quantitative measures such as bit-rates, segment lengths, screen size, and so on. Profiles defined in this specification define restrictions on features of this specification, but may additionally impose restrictions on other aspects of media delivery.

NOTE A profile can also be understood as permission for 3GP-DASH clients that only implement the features required by the profile to process the Media Presentation. However, as 3GP-DASH client operation is not specified normatively, it is also unspecified how a 3GP-DASH client conforms to a particular profile. Hence, profiles merely specify restrictions on MPD and Segments rather than DASH client behaviour.

A profile has an identifier, which is a URI. The profiles with which a Media Presentation complies are indicated in the **`MPD`**`@profiles` attribute. This element is a comma-separated list of profile identifiers. Profile identifiers defined in this specification are URNs conforming to RFC 3406 [21]. URLs may also be used. When a URL is used, it should also contain a month-date in the form mmyyyy; the assignment of the URL must have been authorized by the owner of the domain name in that URL on or very close to that date, to avoid problems when domain names change ownership.

An MPD is conforming when it satisfies the following:

1. The MPD is valid in terms the schema defined in Annex B.
2. The MPD conforms to the normative requirements defined in this specification.
3. The MPD conforms to each of the profiles indicated in the **`MPD`**`@profiles` attribute as specified below.

When *ProfA* is included in the **`MPD`**`@profiles` attribute, the MPD is modified into a profile-specific MPD for profile conformance checking using the following ordered steps:

1. The **`MPD`**`@profiles` attribute of the profile-specific MPD contains only *ProfA*.
2. An **`AdaptationSet`** element for which `@profiles` does not or is not inferred to include *ProfA* is removed from the profile-specific MPD.
3. A **`Representation`** element for which `@profiles` does not or is not inferred to include *ProfA* is removed from the profile-specific MPD.
4. All elements or attributes that are either (i) in this specification and explicitly excluded by *ProfA*, or (ii) in an extension namespace and not explicitly included by *ProfA*, are removed from the profile-specific MPD.
5. All elements and attributes that 'may be ignored' according to the specification of *ProfA* are removed from the profile-specific MPD,

An MPD is conforming to profile *ProfA* when it satisfies the following:

1. *ProfA* is included in the **`MPD`**`@profiles` attribute.
2. The profile-specific MPD for *ProfA* is valid in terms the schema defined in Annex B.
3. The profile-specific MPD for *ProfA* conforms to the normative semantics defined in this specification.
4. The profile-specific MPD for *ProfA* conforms to the restrictions specified for *ProfA*.

A Media Presentation is conforming to profile *profA* when it satisfies the following:

1. The MPD of the Media Presentation is conforming to profile *ProfA* as specified above.
2. There is at least one Representation in each Period in the profile-specific MPD for *ProfA*.
3. The Segments of the Representations of the profile-specific MPD for *ProfA* conform to the restrictions specified for *ProfA*.

NOTE In other words, each MPD contains at least one Representation in each Period, which fulfils the requirements of a profile listed in **MPD**@profiles. There may be stricter rules on the occurrence of Representations in the specified profiles. For example, it can be required that there is at least one Representation for each media type that contains or is inferred to have the profile identifier of a specific profile.

7.3.2 3GPP Adaptive HTTP Streaming (Release-9 AHS)

Release-9 Adaptive HTTP Streaming as defined in TS 26.234 [3] Release-9, clause 12 is not a profile of this specification. Rel-9 AHS uses a different namespace "urn:3GPP:ns:PSS:AdaptiveHTTPStreamingMPD:2009" and a different MIME type signalling "application/3gpp-ahs+xml" for the MPD. However, a Media Presentation may be defined such that segments complying with the segment formats in TS 26.234 [3] Release-9, clause 12, also comply with segment formats for this specification.

7.3.3 3GP-DASH Release-10 Profile

7.3.3.1 Introduction

The 3GP-DASH Release-10 profile is identified by the URN 'urn:3GPP:PSS:profile:DASH10'.

This profile includes all features defined in the Release-10 version of this specification in clauses 7.3.6 (media codecs), 7.3.7 (content protection), 8 (Media Presentation Description), 9 (File Format) and 10 (QoE).

The @mimeType attribute of each Representation shall be provided according to RFC4337. Additional parameters may be added according to RFC6381 [26].

7.3.4 Void

7.3.5 Void

7.3.6 Media Codecs

For 3GP-DASH clients supporting a particular continuous media type, media decoders are specified in TS 26.234 [3], clause 7.2 for speech, 7.3 for audio, 7.4 for video, 7.9 for timed text and 7.11 for timed graphics.

7.3.7 Content Protection

3GP-DASH clients content protection may support OMA DRM 2.0 [15] or OMA DRM 2.1 [16]. Other content protection schemes may be supported. The ContentProtection element in the MPD should be used to convey content protection information.

When using OMA DRM V2.0 or OMA DRM V2.1 scheme for content protection, the non-streamable Packetized DRM Content Format (PDCF) shall be used. An OMA-DRM encrypted Representation shall include the brand 'opf2'. OMA-DRM [15] [16] defines the procedures for acquiring the Rights Object from the Rights Issuer to decrypt PDCF protected content. The scheme is identified by a **ContentProtection**@schemeIdUri set to "urn:mpeg:dash:mp4protection" and the **ContentProtection**@value shall include the version number; it starts with "odkm", which is the scheme_type contained in the Scheme Type Box of the PDCF file, followed by a ":" and the scheme_version from the Scheme Type Box of the PDCF file, encoded as up to 8 hexadecimal digits, where the leading "0"s may be omitted. For example, for OMA DRM2.0 the value could be "odkm:200".

8 DASH - Media Presentation

8.1 Introduction

A Media Presentation is a structured collection of data that is accessible to a 3GP-DASH client to provide a streaming service to the user.

3GP-DASH is intended to support a media-streaming model for delivery of media content in which control of the delivery lies exclusively with the client. Clients may request data using the HTTP protocol from standard web servers that have no 3GP-DASH-specific capabilities. Consequently, this standard focuses not on client or server procedures but on the data formats used to provide a DASH Media Presentation.

The collection of encoded and deliverable versions of media content and the appropriate description of these form a Media Presentation. Media content is composed of a single or multiple contiguous media content **periods** in time. Each media content period is composed of one or multiple **media content components,** for example audio components in various languages and a video component. Each media content component has an assigned **media content component type**, for example audio or video.

Each media content component may have several encoded versions, referred to as **media streams**. Each media stream inherits the properties of the media content, the media content period, the media content component from which it was encoded and in addition it gets assigned the properties of the encoding process such as sub-sampling, codec parameters, encoding bitrate, etc. This describing metadata is relevant for static and dynamic selection of media content components and media streams.

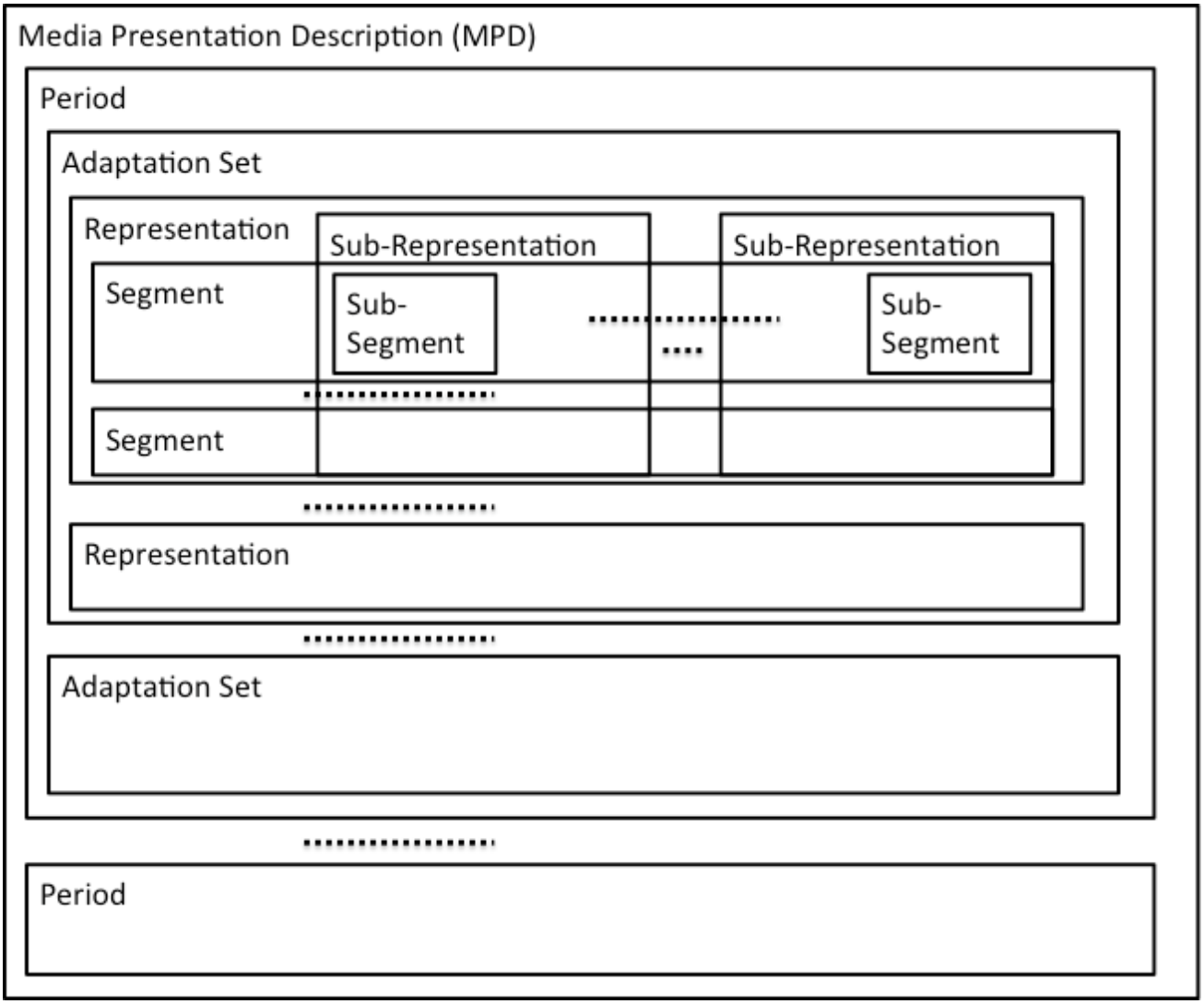

Figure 8.1: 3GP-DASH High-Level Data Model

DASH is based on a hierarchical data model aligned with the presentation in Figure 8.1. A DASH Media Presentation is described by a **Media Presentation Description** (see clause 8.4.1) document. This describes the sequence of **Periods** (see clause 8.4.2) in time that make up the Media Presentation. A Period typically represents a media content period during which a consistent set of encoded versions of the media content is available i.e. the set of available bitrates, languages, captions, subtitles etc. does not change during a Period.

Within a Period, material is arranged into **Adaptation Sets** (see clause 8.4.3.3). An Adaptation Set represents a set of interchangeable encoded versions of one or several media content components. For example there may be one Adaptation Set for the main video component and a separate one for the main audio component. If there is other material available, for example captions or audio descriptions, then these may each have a separate Adaptation Set. Material may also be provided in multiplexed form, in which case interchangeable versions of the *multiplex* may be described as a single Adaptation Set, for example an Adaptation Set containing both the main audio and main video for a Period. Each of the multiplexed components may be described individually by a media content component description.

An Adaptation Set contains a set of **Representations** (see clause 8.4.3.4). A Representation describes a *deliverable encoded version* of one or several media content components. A Representation includes one or more media streams (one for each media content component in the multiplex). Any single Representation within an Adaptation Set is sufficient to render the contained media content components. Typically, clients may switch from Representation to Representation within an Adaptation Set in order to adapt to network conditions or other factors. Clients may also ignore Representations that rely on codecs or other rendering technologies they do not support or that are otherwise unsuitable.

Within a Representation, the content may be divided in time into **Segments** (see clause 8.4.4 and clause 9). A URL is provided for each Segment meaning that a Segment is the largest unit of data that can be retrieved with a single HTTP request.

DASH defines different timelines. One of the key features in DASH is that encoded versions of different media content components share a common timeline. The presentation time of access unit within the media content is mapped to the global common presentation timeline for synchronization of different media components and to enable seamless switching of different coded versions of the same media components. This timeline is referred as Media Presentation timeline. The Media Segments themselves contain accurate Media Presentation timing information enabling synchronization of components and seamless switching.

A second timeline is used to signal to clients the availability time of Segments at the specified HTTP-URLs. These times are referred to as **Segment availability times** and are provided in wall-clock time. Clients typically compare the wall-clock time to Segment availability times before accessing the Segments at the specified HTTP-URLs. For On-Demand services with a static MPD, the availability times of all Segments are identical. For live services when the MPD is updated, the availability times of Segments depend on the position of the Segment in the Media Presentation timeline.

Segments are assigned a duration, which is 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.

Short Segments are usually required in the case of live content, where there are restrictions on end-to-end latency. The duration of a Segment is typically a lower bound on the end-to-end latency. DASH does not support the possibility for Segments to be extended over time: a Segment is a complete and discrete unit that must be made available in its entirety.

Segments may be further subdivided into **Subsegments** each of which contains a whole number of complete access units. In formats defined in this specification, a Subsegment contains a whole number of complete movie fragments. A Segment may be divided into Subsegments described by a compact **Segment index**, which provides the presentation time range in the Representation and corresponding byte range in the Segment occupied by each Subsegment. Clients may download this index in advance and then issue requests for individual Subsegments.

Clients may switch from Representation to Representation within an Adaptation Set at any time in the media content. However, switching at arbitrary positions may be complicated because of coding dependencies within Representations and other factors. It is also desirable to avoid download of 'overlapping' data i.e. media for the same time period from multiple Representations. Usually, switching is simplest at a Stream Access Point in the new stream. In order to formalize requirements related to switching DASH defines a codec-independent concept of Stream Access Points and identifies various types of Stream Access Points.

Segmentation and Subsegmentation may be performed in ways that make switching simpler. For example, in the very simplest cases each Segment or Subsegment begins with a Stream Access Point and the boundaries of Segments or Subsegments are aligned across the Representations of an Adaptation Set. In this case, switching Representation involves playing to the end of a (Sub)Segment of one Representation and then playing from the beginning of the next (Sub)Segment of the new Representation. The Media Presentation Description and Segment Index provide various indications, which describe properties of the Representations that may make switching simpler.

For On-Demand services, the Media Presentation Description is a static document describing the various aspects of the Media Presentation. All Segments of the Media Presentation are available on the server once any Segment is available. For live services, however, Segments become available with time as the content is produced. The Media Presentation Description may be updated regularly to reflect changes in the presentation over time, for example Segment URLs for new Segments may be added to the MPD and those for old, no longer available Segments may be removed. However, if Segment URLs are described using a template, this updating may not be necessary except for some redundancy/failover cases.

In summary a Media Presentation is described in a Media Presentation Description (MPD) including any possible updates of the MPD. The MPD is defined in clause 8.2 and the update mechanisms in 8.5. Assembly of a fragmented MPD is defined in 8.3. The data model that constitutes a Media Presentation is defined in 8.4 and some additional elements in the MPD that describe the content are provided in 8.6.

8.2 Media Presentation Description

8.2.1 General

The Media Presentation Description (MPD) is a document that contains metadata required by a 3GP-DASH client to construct appropriate HTTP-URLs to access Segments and to provide the streaming service to the user.

NOTE: actual playback of the media streams included in the Representations is not controlled by the MPD information. Playback is controlled by the media engine operating on the media streams contained in the Representations in the usual way.

The format of URLs in the MPD and the process to generate HTTP GET and partial GET requests from URLs provided in the MPD is defined in 8.7.

The MPD is an XML-document that is formatted according to the XML schema provided in clause 8.2.2.

The MIME type of the MPD shall be '`application/dash+xml`' as defined in Annex H.1.

The encoding of the MPD shall be UTF-8 as defined in IETF RFC 3629 [34]. All data provided in extension namespaces shall be UTF-8 as defined in IETF RFC 3629 [34]. If binary data needs to be added, it shall be included in Base64 as described in IETF RFC 4648 [36] within a UTF-8 encoded element with a proper name space or identifier, such that an XML parser knows how to process or ignore it.

MPDs may be updated as specified in clause 8.5. Updates may also be done using MPD delta files as defined in clause 8.5.2. The MIME type of an MPD delta file shall be '`application/dashdelta`' as defined in Annex H.2.

The delivery of the MPD is not in scope of this specification. If the MPD is delivered over HTTP, then the MPD may be transfer encoded for transport, as described in [18] using the generic GZip algorithm RFC 1952 [18]. 3GP-DASH clients shall support GZip content decoding of the MPD when delivered over HTTP (GZIP RFC 1952 [18], clause 9).

8.2.2 Schema and 3GPP Extension

The overview of the XML schema of the MPD is provided in below. Specific types, elements and attributes are introduced in the remainder of this clause. The complete MPD schema is provided in Annex B of this specification. In case of any inconsistencies the schema in Annex B takes precedence over the XML-syntax snippets provided in this clause. For the normative schema refer to the schema in Annex B.

The main schema is provided in Table 8-1 with the namespace "`urn:mpeg:dash:schema:mpd:2011`". The 3GPP extension namespace is provided in Table 8-2 with namespace "`urn:3GPP:ns:DASH:MPD-ext:2011`". An extension schema for 3GPP in the context of the specification is referred to as "3gpp-2011.xsd". Elements and attributes in the extension namespace are preceded with "`x3gpp:`" throughout this document.

The MPD shall be authored such that, after unrecognized XML attributes or elements are removed, the result is a valid XML document formatted according to the XML schema provided in Annex B and that complies with this specification. Namespaces may be used to extend functionalities. Therefore, all extended elements and attributes added to a **`Representation`** in particular shall be such that they can be safely ignored by 3GP-DASH clients.

Example for valid MPDs are provided in Annex D.

Table 8-1: Overview of XML schema of the MPD

```
<?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:x3gpp="urn:3GPP:ns:DASH:MPD-ext:2011"
    xmlns="urn:mpeg:dash:schema:mpd:2011">
```

```
    <xs:annotation>
        <xs:appinfo>Media Presentation Description</xs:appinfo>
    </xs:annotation>

    <xs:import namespace="http://www.w3.org/1999/xlink" schemaLocation="xlink.xsd"/>
    <xs:import namespace="urn:3GPP:ns:DASH:MPD-ext:2011" schemaLocation="3gpp-2011.xsd"/>

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

</xs:schema>
```

Table 8-2: Overview of XML schema for 3GPP MPD extensions

```
<?xml version="1.0"?>
<xs:schema targetNamespace="urn:3GPP:ns:DASH:MPD-ext:2011"
    attributeFormDefault="unqualified" elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="urn:3GPP:ns:DASH:MPD-ext:2011">
    <xs:annotation>
        <xs:appinfo>Extensions to Media Presentation Description for 3GPP</xs:appinfo>
    </xs:annotation>

    ...

</xs:schema>
```

8.2.3 (void)

8.2.4 (void)

8.3 MPD Assembly

8.3.1 Introduction

This clause defines a mechanism for referencing a remote DASH element from within a local MPD. A subset of W3C XLINK [20] simple links is defined consisting of:

- restricted syntax and semantics in clause 8.3.2, and
- the processing model in clause 8.3.3.

8.3.2 Syntax and semantics

Table 8-3 provides the XLINK attributes that are used in this specification and shall be supported accordingly.

Table 8-3: XLINK attributes used in this specification

Attribute	Comments and Usage
`@xlink:type`	Identifies the type of W3C XLINK being used. In the context of specification, all references shall be W3C XLINK simple links. As the attribute `@xlink:type` is optional with fixed setting `@xlink:type="simple"`.
`@xlink:href`	Identifies the remote DASH Element by URI as defines in IETF RFC 3986 [17]. In the context of this specification, URI shall exclusively be HTTP-URLs.
`@xlink:show`	Defines the desired behaviour of a remote DASH element once dereferenced from within a MPD as defined in W3C XLINK. In the context of this specification the attribute `@xlink:show` is optional with fixed setting `@xlink:show="embed"`. NOTE: In W3C XLINK, the behaviour of conforming XLink applications when embedding XML-based ending resources, such as a remote DASH element, is not defined. Thus, the actual behaviour for this standard is defined in clause 8.3.3.

Attribute	Comments and Usage
`@xlink:actuate`	Defines the desired timing of dereferencing a remote DASH-Element from within a MPD as defined in W3C XLINK. The following attribute values are allowed in this standard: 1) `onLoad`: an application should dereference the remote DASH element immediately on loading the MPD. 2) `onRequest` (default): formally, an application should dereference the remote DASH-element only on a post-loading event triggered for the purpose of dereferencing. In the context of this specification, the application dereferences the link only for those resources it needs (or anticipates it probably will need). Examples include de-referencing a link in a **`Period`** element when the play-time is expected to enter that period, de-referencing a representation group link when it appears to contain representations that will be needed, and so on.

The restricted schema for XLINK in the context of the standard is referred to as "xlink.xsd" in any schema in this standard and defined in Table 8-4.

Table 8-4: XML Schema for XLINK attributes used in this specification

```
<?xml version='1.0'?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.w3.org/1999/xlink"
    xmlns:xlink="http://www.w3.org/1999/xlink">

    <xs:attribute name="type" type="xs:token" fixed="simple"/>

    <xs:attribute name="href" type="xlink:hrefType"/>

    <xs:simpleType name="hrefType">
        <xs:restriction base="xs:anyURI"/>
    </xs:simpleType>

    <xs:attribute name="show" type="xs:token" fixed="embed"/>

    <xs:attribute name="actuate" type="xlink:actuateType" default="onRequest"/>

    <xs:simpleType name="actuateType">
        <xs:restriction base="xs:token">
            <xs:enumeration value="onLoad"/>
            <xs:enumeration value="onRequest"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

8.3.3 Processing

The following rules apply to the processing of URI references within `@xlink:href`:

1) URI references to remote elements that cannot be resolved shall be treated as invalid references and invalidate the MPD.

2) Only a single element type shall be included in a remote element. However, multiple elements of the same type may be included in a remote element unless explicitly restricted. If multiple root elements are obtained from the remote element, the elements shall be returned in the appropriate order and the first element shall replace the element within the MPD, using the rules defined above. All other elements shall be inserted immediately after this element in the order in which they are declared.

3) URI references to remote elements that are inappropriate targets for the given reference shall be treated as invalid references (see list below for the appropriate targets) references and invalidate the MPD.

4) URI references that directly or indirectly reference themselves are treated as invalid circular references references and invalidate the MPD.

5) If a URI reference is relative then reference resolution as defined in 8.7.4 shall apply.

The remote elements referenced from within an MPD (referred to as appropriate targets) shall be embedded into the MPD by applying the following rules:

1) Attributes and elements obtained from the remote element shall be added to the element of the MPD that contains `@xlink:href` and shall be merged with the ones already present in the MPD. If the same attributes are present in both MPD and remote element, the attribute values should be the same. If they are not identical, then the value of the attribute of the MPD takes precedence over the value of the attribute in the remote DASH element.

2) The remote DASH element referenced by the `@xlink:href` shall conform to the type definition of the element in the MPD that contains `@xlink:href`.

3) All XLINK attributes shall be removed after dereferencing is completed.

4) Only a single element type shall be included in a remote DASH element. However, multiple elements of the same type may be included in a remote element unless explicitly restricted.

5) All resources in the remote element referenced by `@xlink:href` shall have an availability end time as specified by **`MPD`**`@availabilityEndTime`.

8.4 Hierarchical Data Model

8.4.1 General

A Media Presentation is described in the **`MPD`** element that is contained in an MPD document formatted as defined in clause 8.2.

A Media Presentation consists of:

- A sequence of one or more Periods described in 8.4.2.
- Each Period contains one or more Adaptation Sets that itself contains one or more Representations as described in clause 8.4.3. Clause 8.4.3 also defines media content components and Sub-Representations.
- Each Representation consists of one or more Segments. Segment Information is introduced in clause 8.4.4. Segments contain media data and/or metadata to access, decode and present the included media content.

The summary of the semantics of the attributes and elements within an **`MPD`** element are provided in Table 8-5. The XML-syntax of the **`MPD`** element is provided in Table 8-6.

Table 8-5: Semantics of `MPD` element

Element or Attribute Name	Use	Description
`MPD`		The root element that carries the Media Presentation Description for a Media Presentation.
`@id`	O	specifies an identifier for the Media Presentation. It is recommended to use an identifier that is unique within the scope in which the Media Presentation is published. If not specified, no MPD-internal identifier is provided. However, for example the URL to the MPD may be used as an identifier for the Media Presentation.
`@profiles`	M	specifies a list of Media Presentation profiles as described in section 7.3. The contents of this attribute shall conform to either the `pro-simple` or `pro-fancy` productions of RFC6381, Section 4.5, without the enclosing `DQUOTE` characters, i.e. including only the `unencodedv` or `encodedv` elements respectively. As profile identifier the URI defined for the conforming Media Presentation profiles as described in 7.3 shall be used.
`@type`	OD default: `static`	specifies whether the Media Presentation Description may be updated (`@type="dynamic"`) or not (`@type="static"`).

Element or Attribute Name	Use	Description
		NOTE Static MPDs are typically used for On-Demand services, whereas dynamic MPDs are used for live services.
@availabilityStartTime	CM Must be present for type='Live'	For @type='dynamic' this attribute shall be present. In this case it specifies the anchor for the computation of the segment availability start time for any Segment in the Media Presentation. For @type='static' , if present, it specifies the Segment availability start time for all Segments referred to in this MPD. If not present, all Segments described in the MPD shall be become available at the time the MPD becomes available.
@availabilityEndTime	O	specifies the latest Segment availability end time for any Segment in the Media Presentation. When not present, the value is unknown.
@mediaPresentationDuration	CM Must be present for @type='static'	specifies the duration of the entire Media Presentation. If the attribute is not present, the duration of the Media Presentation is unknown. In this case the attribute **MPD**@minimumUpdatePeriod shall be present. This attribute shall be present when the attribute **MPD**@minimumUpdatePeriod is not present.
@minimumUpdatePeriod	O	If this attribute is present, it specifies the smallest period between potential changes to the MPD. This can be useful to control the frequency at which a client checks for updates. If this attribute is not present it indicates that the MPD does not change. If **MPD**@type is "static", @minimumUpdatePeriod shall not be present. Details on the use of the value of this attribute are specified in 8.5.
@minBufferTime	M	specifies a common duration used in the definition of the Representation data rate (see @bandwidth attribute in 8.4.3.4).
@timeShiftBufferDepth	O	specifies the duration of the smallest time shifting buffer for any Representation in the MPD that is guaranteed to be available for a Media Presentation with type 'dynamic'. When not present, the value is infinite. This value of the attribute is undefined if the @type attribute is equal to "static".
@suggestedPresentationDelay	O	when @type is 'dynamic', it specifies a fixed delay offset in time from the from the presentation time of each access unit that is suggested to be used for presentation of each access unit. For more details refer to 9.4.1.2. When not specified, then no value is provided and the client is expected to choose a suitable value. when @type is 'static' the value of the attribute is undefined and may be ignored.
@maxSegmentDuration	O	specifies the maximum duration of any Segment in any Representation in the Media Presentation, i.e. documented in this MPD and any future update of the MPD. If not present, then the maximum Segment duration shall be the maximum duration of any Segment documented in this MPD.
@maxSubsegmentDuration	O	specifies the maximum duration of any Media Subsegment in any Representation in the Media Presentation. If not present, the same value as for the maximum Segment duration is implied.
ProgramInformation	0 … N	specifies descriptive information about the program. For more details refer to the description in clause 8.6.2.
BaseURL	0 … N	specifies a Base URL that can be used for reference resolution and alternative URL selection. For more details refer to the description in clause 8.7.

Element or Attribute Name	Use	Description
x3gpp:DeltaSupport	0 … 1	If present, this element specifies that MPD delta files are supported by the server. For more details refer to the description in clause 8.5.2.
Location	0 … N	specifies an absolute URL where the MPD is available.
Period	1 … N	specifies a Period. For more details refer to the description in clause 8.4.2.
Metrics	0 ... N	specifies information about the requested QoE metrics. For more details refer to clause 10.3. At most one **Metrics** element shall be present in the MPD. NOTE: The schema allows more than one **Metrics** element for potential future extensions.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>…<maxOccurs> (N=unbounded) Elements are **bold**; attributes are non-bold and preceded with an @		

Table 8-6: Syntax of MPD element

```
    <!-- 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 name="x3gpp:DeltaSupport" type="DeltaSupportType" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="id" type=xs:string"/>
        <xs:attribute name="profiles" type="xs:string" use="required"/>
        <xs:attribute name="type" type="PresentationType" default="static"/>
        <xs:attribute name="availabilityStartTime" type="xs:dateTime"/>
        <xs:attribute name="availabilityEndTime" type="xs:dateTime"/>
        <xs:attribute name="mediaPresentationDuration" type="xs:duration"/>
        <xs:attribute name="minimumUpdatePeriod" type="xs:duration"/>
        <xs:attribute name="minBufferTime" type="xs:duration" use="required"/>
        <xs:attribute name="timeShiftBufferDepth" type="xs:duration"/>
        <xs:attribute name="suggestedPresentationDelay" type="xs:duration"/>
        <xs:attribute name="maxSegmentDuration" type="xs:duration"/>
        <xs:attribute name="maxSubsegmentDuration" type="xs:duration"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>


    <!-- Type of presentation - live or on-demand -->
    <xs:simpleType name="PresentationType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="static"/>
            <xs:enumeration value="dynamic"/>
        </xs:restriction>
    </xs:simpleType>
```

8.4.2 Period

A Media Presentation consists of one or more Periods. A Period is defined by **Period** element in the **MPD** element.

The type of the Period, either a regular Period or an Early Available Period, as well as the *PeriodStart* time of a regular Period is determined as follows:

- If the attribute `@start` is present in the **Period**, then the Period is a regular Period and the *PeriodStart* is equal to the value of this attribute.
- If the `@start` attribute is absent, but the previous **Period** element contains a `@duration` attribute then this new Period is also a regular Period. The start time of the new Period *PeriodStart* is the sum of the start time of the previous Period *PeriodStart* and the value of the attribute `@duration` of the previous Period.

- If (i) `@start` attribute is absent, and (ii) the **`Period`** element is the first in the MPD, and (iii) the **`MPD`**`@type` is `'static'`, then the *PeriodStart* time shall be set to zero.

- If (i) `@start` attribute is absent, and (ii) the previous **`Period`** element does not contains a `@duration` attribute or the **`Period`** element is the first in the MPD, and (iii) the **`MPD`**`@type` is `'dynamic'`, then this Period is an Early Available Period (see below for details).

For any regular Period the following holds: *PeriodStart* reflects the actual time that should elapse after playing the media of all prior Periods in this Media Presentation relative to the *PeriodStart* time of the first Period in the Media Presentation. The Period extends until the *PeriodStart* of the next Period, or until the end of the Media Presentation in the case of the last Period. More specifically, the difference between the *PeriodStart* time of a Period and either the *PeriodStart* time of the following Period, if this is not the last Period, or the value of the **`MPD`**`@mediaPresentationDuration` if this is the last one, is the presentation duration in Media Presentation time of the media content represented by the Representations in this Period.

Early Available Periods may be used to advertise initialization of other non-media data before the media data itself is available. **`Period`** elements documenting early available Periods shall not occur before any **`Period`** element documenting a regular Period. For Early Available Periods, any resources that are announced in such a **`Period`** element shall be available. Such a **`Period`** element shall not contain URLs to Media Segments. The data contained in such a **`Period`** element does not represent a Period in the Media Presentation. Only when the *PeriodStart* time becomes known through an update of the MPD, such a **`Period`** element represents a regular Period. However, an update of the MPD may even remove a **`Period`** element representing an Early Available Period in later updates of the MPD as long as no *PeriodStart* time is associated with the Period.

The attributes and elements contained in the **`Period`** element are provided in Table 8-7 along with their semantics. The XML syntax or the **`Period`** element is provided in Table 8-8.

Table 8-7: Semantics of `Period` Element

Element or Attribute Name	Use	Description
`Period`		specifies the information for a single Period.
`@xlink:href`	O	specifies a reference to a remote element that contains one or multiple elements of type **`Period`**
`@xlink:actuate`	O default: `onRequest`	specifies the processing instructions, which can be either "`onLoad`" or "`onRequest`". This attribute must not be present if the `@xlink:href` attribute is not present
`@id`	CM if "**`MPD`**`@type=` `dynamic"`	specifies a unique identifier for this Period within the Media Presentation. The identifier shall be unique within the scope of the Media Presentation. If the **`MPD`**`@type` is equal to "`dynamic`", then this attribute shall be present and the `@id` of the Period shall not change in case an MPD is updated. If not present, no identifier for the Period is provided.
`@start`	O	if present, specifies the *PeriodStart* time of the Period. The *PeriodStart* time is used as an anchor to determine the MPD start time of each Media Segment as well as to determine the presentation time of each each access unit in the Media Presentation timeline. If not present, refer to the details above in this clause.
`@duration`	O	if present, specifies the duration of the Period to determine the *PeriodStart* time of the next Period. If not present, refer to the details above in this clause.
`@bitstreamSwitching`	OD Default: `false`	When set to "`true`", this is equivalent as if the **`AdaptationSet`**`@bitstreamSwitching` for each Adaptation Set contained in this Period is set to '`true`'. In this case, the **`AdaptationSet`**`@bitstreamSwitching` attribute shall not be set to '`false`' for any Adaptation Set in this Period.

Element or Attribute Name	Use	Description
BaseURL	0…N	specifies a base URL that can be used for reference resolution and alternative URL selection. For more details refer to the description in clause 8.7.
SegmentBase	0...1	specifies default Segment Base information. Information in this element is overridden by information in **AdaptationSet.SegmentBase** and **Representation.SegmentBase**, if present. For more details see clause 8.4.4.
SegmentList	0...1	specifies default Segment List information. Information in this element may be overridden by information in **AdaptationSet.SegmentList** and **Representation.SegmentList**, if present. For more details see clause 8.4.4.
SegmentTemplate	0...1	specifies default Segment Template information. Information in this element may be overridden by information in **AdaptationSet.SegmentTemplate** and **Representation.SegmentTemplate**, if present. For more details see clause 8.4.4.
AdaptationSet	0...N	specifies an Adaptation Set. At least one Adaptation Set shall be present in each Period unless the value of the `@duration` attribute of the Period is set to zero. Note that the actual element may be present only in a remote element if xlink is in use. For more details see clause 8.4.3.3.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Note that the conditions only holds without using xlink:href. If linking is used, then all attributes are "optional" and <minOccurs=0> Elements are **bold**; attributes are non-bold and preceded with an @.		

Table 8-8: Syntax of `Period` Element

```
<!-- Period of a presentation -->
<xs:complexType name="PeriodType">
    <xs:sequence>
        <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="SegmentBase" type="SegmentBaseType" minOccurs="0"/>
        <xs:element name="SegmentList" type="SegmentListType" minOccurs="0"/>
        <xs:element name="SegmentTemplate" type="SegmentTemplateType" minOccurs="0"/>
        <xs:element name="AdaptationSet" type="AdaptationSetType" minOccurs="0"
            maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute ref="xlink:href"/>
    <xs:attribute ref="xlink:actuate" default="onRequest"/>
    <xs:attribute name="id" type="xs:string"/>
    <xs:attribute name="start" type="xs:duration"/>
    <xs:attribute name="duration" type="xs:duration"/>
    <xs:attribute name="bitStreamSwitching" type="xs:boolean" default="false"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

8.4.3 Adaptation Sets and Representations

8.4.3.1 Overview

Periods are further subdivided as follows:

- Each Period contains one or more groups. Groups consist of Adaptation Sets as described in clause 8.4.3.3.
- In case an Adaptation Set contains multiple media content components, then each media content component is described individually as defined in clause 8.4.3.6.
- Each Adaptation Set contains one or more Representations as described in clause 8.4.3.4.
- A Representation may contain one or more Sub-Representations as described in clause 8.4.3.5.
- Adaptation Sets, Representations and Sub-Representations share common attributes and elements that are described in clause 8.4.3.2.

8.4.3.2 Common Attributes and Elements

The elements **AdaptationSet, Representation** and **SubRepresentation** have assigned common attributes and elements.

The attributes and elements listed in Table 6 may be present in all three elements. The semantics of these attributes are provided in Table 8-9. The XML-syntax is provided in Table 8-10.

The 'Use' column in Table 8-9 shall be interpreted that an attribute marked with 'M' shall be available for a Representation, i.e. it shall either be present in the **Representation** element, or if not, it shall be in the containing **AdaptationSet** element. An attribute marked with 'O' may be absent in both.

AdaptationSet element. An attribute marked with 'O' may be absent in both.

Table 8-9: Common Adaptation Set, Representation and Sub-Representation and Attributes and Elements

Element or Attribute Name	Use	Description
Common attributes and elements		
@profiles	O	specifies the profiles which the associated Representation(s) conform to the list of Media Presentation profiles as described in 7.3. The value shall be a subset of the respective value in any higher level of the document hierarchy (Representation, Adaptation Set, MPD). If not present, the value is inferred to be the same as in the next higher level of the document hierarchy. For example, if the value is not present for a Representation, then @profiles at the Adaptation Set level is valid for the Representation. The same syntax as defined in 8.4.1 shall be used.
@width	O	Specifies the horizontal visual presentation size of the video media type in pixel. If not present on any level, the value is unknown.
@height	O	Specifies the vertical visual presentation size of the video media type in pixel. If not present on any level, the value is unknown.
@frameRate	O	specifies the output frame rate of the video media type in the Representation. If the frame rate is varying, the value is the average frame over the entire duration of the Representation. The value is coded as a string, either containing two integers separated by a "/", ("F/D"), or a single integer "F". The frame rate is the division F/D, or F, respectively, per second (i.e. the default value of D is '1'). If not present on any level, the value is unknown.
@audioSamplingRate	O	Either a single decimal integer value specifying the sampling rate or a whitespace separated pair of decimal

Element or Attribute Name	Use	Description
		integer values specifying the minimum and maximum sampling rate of the audio media component type. The values are in samples per second. If not present on any level, the value is unknown.
@mimeType	M	specifies the MIME type of the concatenation of the Initialization Segment, if present, and all consecutive Media Segments in the Representation.
@codecs	M	specifies the codecs parameter specifying the media types. The codec parameters shall also include the profile and level information where applicable. The contents of this attribute shall conform to either the simp-list or fancy-list productions of RFC6381 [26] clause 3.2, without the enclosing DQUOTE characters. The codec identifier for the media format, mapped into the name space for codecs as specified in RFC6381 [26], clause 3.3 shall be used.
@maximumSAPPeriod	O	when present, specifies the maximum SAP interval in seconds of all contained media streams, where the SAP interval is the maximum time interval between the T_{SAP} of any two successive SAPs of types 1 to 3 inclusive of one media stream in the associated Representations. If not present on any level, the value is unknown.
@startWithSAP	O	when present and greater than 0, specifies that in the associated Representations, each Media Segment starts with a SAP of type less than or equal to the value of this attribute value in each media stream. A Media Segment starts with a SAP in a media stream if the stream contains a SAP in that Media Segment, I_{SAU} is the index of the first access unit that follows I_{SAP} and I_{SAP} is contained in the Media Segment. If not present on any level, the value is unknown.
@maxPlayoutRate	O	Specifies the maximum playout rate as a multiple of the regular playout rate, which is supported with the same decoder profile and level requirements as the normal playout rate. If not present on any level, the value is 1.
@codingDependency	O	When present and "true", for all contained media streams, specifies that there is at least one access unit that depends on one or more other access units for decoding. When present and "false", for any media type, there is no access unit that depends on any other access unit for decoding (e.g. for video all the pictures are intra coded). When not present, there may or may not be coding dependency between access units.
AudioChannelConfiguration	0 … N	specifies the audio channel configuration of the audio media component type. For detaiis see clause 8.6.3.1 and 8.6.3.7.
ContentProtection	0 … N	specifies information about the use of content protection for the associated Representations. For details, refer to clause 8.6.3.1 and 8.6.3.2.
EssentialProperty	0 … N	specifies information about the containing element that is considered essential by the Media Presentation author for processing the containing element. For details see clause 8.6.3.8.
SupplementalProperty	0 … N	specifies supplemental information about the containing element that may be used by the DASH client optimizing the processing. For details see clause 8.6.3.9.

Element or Attribute Name	Use	Description
Legend: For attributes: M=Mandatory, O=Optional. For elements: <minOccurs>..<maxOccurs> (N=unbounded) Elements are **bold**; attributes are non-bold and preceded with an @.		

Table 8-10: XML-Syntax of Common Group and Representation and Attributes and Elements

```
    <!-- RepresentationBase type; extended by other Representation-related types -->
    <xs:complexType name="RepresentationBaseType" abstract="true">
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element name="AudioChannelConfiguration" type="DescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="ContentProtection" type="DescriptorType" minOccurs="0"
                maxOccurs="unbounded"/>
             <xs:element name="EssentialProperty" type="DescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
              <xs:element name="SupplementalProperty" type="DescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="profiles" type="xs:string"/>
        <xs:attribute name="width" type="xs:unsignedInt"/>
        <xs:attribute name="height" type="xs:unsignedInt"/>
        <xs:attribute name="frameRate" type="FrameRateType"/>
        <xs:attribute name="audioSamplingRate" type="xs:string"/>
        <xs:attribute name="mimeType" type="xs:string"/>
        <xs:attribute name="codecs" type="xs:string"/>
        <xs:attribute name="maximumSAPPeriod" type="xs:double"/>
        <xs:attribute name="startWithSAP" type="SAPType"/>
        <xs:attribute name="maxPlayoutRate" type="xs:double"/>
        <xs:attribute name="codingDependency" type="xs:boolean"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

<!-- Stream Access Point type enumeration -->
<xs:simpleType name="SAPType">
  <xs:restriction base="xs:unsignedInt">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="6"/>
  </xs:restriction>
</xs:simpleType>

  <!-- Type for Frame Rate -->
  <xs:simpleType name="FrameRateType">
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9]*[0-9](/[0-9]*[0-9])?"/>
    </xs:restriction>
  </xs:simpleType>
```

8.4.3.3 Adaptation Set

An Adaptation Set is described by an **AdaptationSet** element. **AdaptationSet** elements are contained in a **Period** element. An Adaptation Set contains alternate Representations, i.e. only one Representation within an Adaptation Set is expected to be presented at a time. All Representations contained in one Adaptation Set represent the same media content components and therefore contain media streams that are considered to be perceptually equivalent.

Representations are arranged into Adaptation Sets according to their to the media content component properties of the media content components present in the Representations, namely

- the language as described by the `@lang` attribute,
- the media component type described by the `@contentType` attribute,
- the role property as described by the **Role** elements,
- the accessibility property as described by the **Accessibility** elements,
- the viewpoint property as described by the **ViewPoint** elements,

- the rating property as described by the **Rating** elements.

Representations shall appear in the same Adaptation Set if and only if they have identical values for all of these media content component properties for each media content component.

The values for the elements **Role**, **Accessibility**, **ViewPoint** and **Rating** are typically not provided within this specification. However, a number of simple schemes are defined in Annex C.

If there exist multiple media content components then the properties of each media content component shall be described by a separate **ContentComponent** element as defined in 8.4.3.6. The **ContentComponent** element shares common elements and attributes with the **AdaptationSet** element. Default values or values applicable to all media content components may be provided directly in the **AdaptationSet** element. Attributes present in the **AdaptationSet** shall not be repeated in the **ContentComponent** element.

The **AdaptationSet** element may contain default values for elements and attributes associated to the contained Representations. Any of the common attributes defined in clause 8.4.3.2 shall only be present either in the **AdaptationSet** element or in the **Representation** element, but not in both.

The **AdaptationSet** element also supports the description of ranges for the `@bandwidth`, `@width`, `@height` and `@frameRate` attributes associated to the contained Representations, which provide a summary of all values for all the Representations within this Adaptation Set. The Representations associated with an **AdaptationSet** element shall not contain values outside the ranges documented for that Adaptation Set.

Adaptation Sets may be further arranged into groups using the `@group` attribute. The semantics of this grouping is that the media content within one Period is represented by:

1) either one Representation from group 0, if present,

2) or the combination of at most one Representation from each non-zero group.

If the **AdaptationSet**`@group` attribute is not present then all Representations in this Adaptation Set are assigned to a non-zero group specific to this Adaptation Set.

The semantics of the attributes and elements within an **AdaptationSet** element are provided in Table 8-11. The XML-syntax of the **AdaptationSet** element is provided in Table 8-12.

Table 8-11: Semantics of `AdaptationSet` element

Element or Attribute Name		Use	Description
AdaptationSet			Adaptation Set description
	`@xlink:href`	O	specifies reference to a remote element that shall contain exactly one element of type **AdaptationSet**
	`@xlink:actuate`	OD default: 'onRequest'	specifies the processing instructions, which can be either "onLoad" or "onRequest".
	`@id`	O	specifies unique identifier for this Adaptation Set within the Period. The attribute shall be unique in the scope of the containing Period. The attribute shall not be present in a remote element. If not present, no identifier for the Adaptation Set is specified.
	`@group`	O	specifies an identifier for the group that is unique in the scope of the containing Period. For details refer to the description above in this clause.
	CommonAttributesElements	-	specifies the common attributes and elements (attributes and elements from base type ***RepresentationBaseType***) For details see clause 8.4.3.2.
	`@lang`	O	Declares the language code(s) for this Adaptation Set. The syntax and semantics according to IETF RFC 5646 [13] shall be used.

Element or Attribute Name		Use	Description
			If not present, the language code may be defined for each media component or it may be unknown.
	@contentType	O	specifies the media content component type for this Adaptation Set. A value of the top-level Content-type 'type' value as defined in RFC 4288 [35], section 4 shall be taken. If not present, the media content component type may be defined for each media component or it may be unknown.
	@minBandwidth	O	specifies minimum bandwidth value in all Representations in this Adaptation Set. This value has the same units as the @bandwidth attribute.
	@maxBandwidth	O	specifies maximum bandwidth value in all Representations in this Adaptation Set. This value has the same units as the @bandwidth attribute.
	@minWidth	O	specifies minimum width value in all Representations in this Adaptation Set. This value has the same units as the @width attribute. If not present, the value is unknown.
	@maxWidth	O	specifies maximum width value in all Representations in this Adaptation Set. This value has the same units as the @width attribute. If not present, the value is unknown.
	@minHeight	O	specifies minimum height value in all Representations in this Adaptation Set. This value has the same units as the @height attribute. If not present, the value is unknown.
	@maxHeight	O	specifies maximum height value in all Representations in this Adaptation Set. This value has the same units as the @height attribute. If not present, the value is unknown.
	@minFrameRate	O	specifies minimum frame rate value in all Representations in this Adaptation Set. This value is encoded in the same format as the @frameRate attribute. If not present, the value is unknown.
	@maxFrameRate	O	specifies maximum frame rate value in all Representations in this Adaptation Set. This value is encoded in the same format as the @frameRate attribute. If not present, the value is unknown.
	@segmentAlignment	O	when not set to "false", this specifies that for any two Representations, X and Y, within the same Adaptation Set, the *m*-th Segment of X and the *n*-th Segment of Y are non-overlapping (as defined in section 9.2.5.2) whenever *m* is not equal to *n*. For Adaptation Sets containing Representations with multiple media content components, this attribute value shall be either 'true' or 'false'. For Adaptation Sets containing Representations with a single media content component, when two **AdaptationSet** elements within a Period share the same integer value for this attribute, then for any two Representations, X and Y, within the union of the two Adaptation Sets, the *m*-th Segment of X and the *n*-th Segment of Y are non-overlapping (as defined in 9.2.5.2) whenever *m* is not equal to *n*.

Element or Attribute Name		Use	Description
	@bitStreamSwitching	O	When this flag is set to "true", the following applies: • All Representations in the Adaptation Set shall have the same number *M* of Media Segments; • Let $R_1, R_2, ..., R_N$ be all the Representations within the Adaptation Set. • Let o $S_{i,j}$, for $j > 0$, be the j^{th} Media Segment in the i^{th} Representation (i.e., R_i), and o $S_{i,0}$ be the Initialization Segment in the i^{th} Representation • The sequence of $S_{i(1),0}, S_{i(1),1}, S_{i(2),2}, ..., S_{i(k),k}, ..., S_{i(M),M,}$ wherein any *i(k)* for all *k* values in the range of 1 to *M*, respectively, is an integer value in the range of 1 to *N*, results in a "conforming Segment sequence" as defined in section 9.2.5.3 with the media format as specified in the @mimeType attribute.
	@subsegmentAlignment	OD default: false	If the **@subsegmentAlignment** for an Adaptation Set is set to other than "false", all following conditions shall be satisfied: • Each Media Segment shall be indexed (i.e. it contains a Segment index) • For any two Representations, X and Y, within the same Adaptation Set, the *m*-th Subsegment of X and the *n*-th Subsegment of Y are non-overlapping (as defined in section 9.2.5.2) whenever *m* is not equal to *n*. For Adaptation Sets containing Representations with a single media content component, when two **AdaptationSet** elements within a Period share the same integer value for this attribute, then for any two Representations, X and Y, within the union of the two Adaptation Sets, the *m*-th Subsegment of X and the *n*-th Subsegment of Y are non-overlapping (as defined in section 9.2.5.2) whenever *m* is not equal to *n*.
	@subsegmentStartsWithSAP	OD default: 0	when greater than 0, specifies that each Subsegment with SAP_type greater than 0 starts with a SAP of type less than or equal to the value of @subsegmentStartsWithSAP. A Subsegment starts with SAP when the Subsegment contains a SAP, and for the first SAP, I_{SAU} is the index of the first access unit that follows I_{SAP}, and I_{SAP} is contained in the Subsegment. The semantics of @subsegmentStartsWithSAP equal to 0 are unspecified.
	Accessibility	0 … N	specifies information about accessibility scheme For more details refer to section 8.6.3.1 and 8.6.3.6.
	Role	0 … N	specifies information on role annotation scheme

Element or Attribute Name		Use	Description
			For more details refer to section 8.6.3.1 and 8.6.3.3.
	Rating	0 … N	specifies information on rating scheme. For more details refer to section 8.6.3.1 and 8.6.3.4.
	Viewpoint	0 … N	specifies information on viewpoint annotation scheme. For more details refer to section 8.6.3.1 and 8.6.3.5.
	ContentComponent	0…N	specifies the properties of one media content component contained in this Adaptation Set. For more details refer to section 8.4.3.6.
	BaseURL	0…N	specifies a base URL that can be used for reference resolution and alternative URL selection. For more details refer to the description in section 8.7.
	SegmentBase	0...1	specifies default Segment Base information. Information in this element is overridden by information in the **Representation.SegmentBase**, if present. For more details see section 8.4.4.
	SegmentList	0...1	specifies default Segment List information. Information in this element is overridden by information in the **Representation.SegmentList**, if present. For more details see section 8.4.4.
	SegmentTemplate	0...1	specifies default Segment Template information. Information in this element is overridden by information in the **Representation.SegmentTemplate**, if present. For more details see section 8.4.4.
	Representation	0 … N	specifies a Representation. At least one Representation element shall be present in each Adaptation Set. The actual element may however be part of a remote element. See subclause 8.4.3.4.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory, F=Fixed. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Note that the conditions only holds without using xlink:href. If linking is used, then all attributes are "optional" and <minOccurs=0> Elements are **bold**; attributes are non-bold and preceded with an @, List of elements and attributes is in ***italics bold***			

Table 8-12: XML-Syntax of `AdaptationSet` element

```
    <!-- Group to contain information common to an adaptation set;
        extends RepresentationBaseType -->
    <xs:complexType name="AdaptationSetType">
        <xs:complexContent>
            <xs:extension base="RepresentationBaseType">
                <xs:sequence>
                    <xs:element name="Accessibility" type="DescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element name="Role" type="DescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element name="Rating" type="DescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element name="Viewpoint" type="DescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element name="ContentComponent" type="ContentComponentType" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element name="BaseURL" type="BaseURLType" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element name="SegmentBase" type="SegmentBaseType" minOccurs="0"/>
                    <xs:element name="SegmentList" type="SegmentListType" minOccurs="0"/>
                    <xs:element name="SegmentTemplate" type="SegmentTemplateType" minOccurs="0"/>
```

```
                    <xs:element name="Representation" type="RepresentationType" minOccurs="0"
                         maxOccurs="unbounded"/>
                    <xs:any namespace="##other" processContents="lax" minOccurs="0"
                         maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute ref="xlink:href"/>
                <xs:attribute ref="xlink:actuate" default="onRequest"/>
                <xs:attribute name="id" type="xs:string"/>
                <xs:attribute name="group" type="xs:unsignedInt"/>
                <xs:attribute name="lang" type="xs:language"/>
                <xs:attribute name="contentType" type="xs:string"/>
                <xs:attribute name="minBandwidth" type="xs:unsignedInt"/>
                <xs:attribute name="maxBandwidth" type="xs:unsignedInt"/>
                <xs:attribute name="minWidth" type="xs:unsignedInt"/>
                <xs:attribute name="maxWidth" type="xs:unsignedInt"/>
                <xs:attribute name="minHeight" type="xs:unsignedInt"/>
                <xs:attribute name="maxHeight" type="xs:unsignedInt"/>
                <xs:attribute name="minFrameRate" type="FrameRateType"/>
                <xs:attribute name="maxFrameRate" type="FrameRateType"/>
                <xs:attribute name="segmentAlignment" type="ConditionalUintType" default="false"/>
                <xs:attribute name="subsegmentAlignment" type="ConditionalUintType"
default="false"/>
                <xs:attribute name="subsegmentStartsWithSAP" type="SAPType" default="0"/>
                <xs:attribute name="bitStreamSwitching" type="xs:boolean"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

<!-- Conditional Unsigned Integer (unsignedInt or boolean) -->
<xs:simpleType name="ConditionalUintType">
  <xs:union memberTypes="xs:unsignedInt xs:boolean"/>
</xs:simpleType>
```

8.4.3.4 Representation

Representations are described by the **`Representation`** element. **`Representation`** elements are contained in an **`AdaptationSet`** element.

A Representation is one of the alternative choices of the complete set or subset of media content components comprising the media content during the defined Period.

A Representation starts at the start of the Period *PeriodStart* and continues to the end of the Period, i.e. the start of the next Period or the end of the Media Presentation.

Each Representation includes one or more media streams, where each media stream is an encoded version of one media content component.

A Representation consists of one or more Segments.

Each Representation either shall contain an Initialization Segment or each Media Segment in the Representation shall be self-initializing, i.e. the Media Segment itself conforms to the media type as specified in the `@mimeType` attribute for this Representation.

The concatenation of the Initialization Segment, if present, and all consecutive Media Segments in one Representation shall represent a conforming Segment sequence as defined in section 9.2.5.3 conforming to the media type as specified in the `@mimeType` attribute for this Representation.

If a Representation is offered in a Media Presentation with **`MPD`**`@type='dynamic'`, it is recommended that means to compensate such drift be included. For more details refer to Annex A.8.

The semantics of the attributes and elements within a Representation are provided in Table 8-13. The XML-syntax of the **`Representation`** element is provided in Table 8-14.

Table 8-13: Semantics of `Representation` element

Element or Attribute Name	Use	Description
`Representation`		This element contains a description of a Representation.
`@id`	M	specifies an dentifier for this Representation. The identifier shall be unique within a Period unless the Representation is

		functionally identically to another Representation in the same Period. The identifier shall not contain whitespace characters. If used in the template-based URL construction as defined in section 8.4.4.4, the string shall only contain characters that are permitted within an HTTP-URL according to RFC 3986 [17]..
`@bandwidth`	M	Consider a hypothetical constant bitrate channel of bandwidth with the value of this attribute in bits per second (bps). Then, if the Representation is continuously delivered at this bitrate, starting at any SAP that is indicated either by `@startWithSAP` or by any Segment Index box, a client can be assured of having enough data for continuous playout providing playout begins after `@minBufferTime * @bandwidth` bits have been received (i.e. at time `@minBufferTime` after the first bit is received).
`@qualityRanking`	O	specifies a quality ranking of the Representation relative to other Representations in the Adaptation Set. Lower values represent higher quality content. If not present then the ranking is undefined.
`@mediaStreamStructureId`	O	The attribute may be present for Representations containing video and its semantics are unspecified for any other type of Representations. If present, the attribute `@mediaStreamStructureId` specifies a whitespace-separated list of media stream structure identifier values. If media streams share the same media stream structure identifier value, the media streams shall have the following characteristics: • The media streams have the same number of Stream Access Points of type 1 to 3. • The values of T_{SAP}, T_{DEC}, T_{EPT}, and T_{PTF} of the *i*-th SAP of type 1 to 3 in one media stream are identical to the values of T_{SAP}, T_{DEC}, T_{EPT}, and T_{PTF}, respectively, of the *i*-th SAP of type 1 to 3 in the other media streams for any value of *i* from 1 to the number of SAPs of type 1 to 3 in any of the media streams. • A media stream formed by concatenating the media stream of a first Representation until I_{SAU} (exclusive) of the *i*-th SAP of type 1 to 3 and the media stream of a second Representation (having the same media stream structure identifier value as for the first Representation) starting from the I_{SAU} (inclusive) of the *i*-th SAP of type 1 to 3 conforms to the specification in which the media stream format is specified for any value of *i* from 1 to the number of SAPs of type 1 to 3 in either media stream. Furthermore, the decoded pictures have an acceptable quality regardless of type of the Stream Access Point access unit used. All media stream structure identifier values for one Adaptation Set shall differ from those of another Adaptation Set. If not present, then for this Representation no similarities to other Representations are known. Note: Indicating multiple media stream structure identifier values for a Representation can be useful in

		cases where switching between Representations A and B as well as between Representations B and C is allowed at non-IDR intra pictures, but switching between Representations A and C would cause too severe a degradation in the quality of the leading pictures and is hence not allowed. To indicate these permissions and restrictions, Representation A would contain `@mediaStreamStructureId` equal to '1', Representation B would contain `@mediaStreamStructureId` equal to '1 2', and Representation C would contain `@mediaStreamStructureId` equal to '2'
CommonAttributesElements	-	Common Attributes and Elements (attributes and elements from base type ***RepresentationBaseType***), for more details see clause 8.4.3.2.
BaseURL	0…N	specifies a Base URL that can be used for reference resolution and alternative URL selection. For more details refer to the description in section 8.7.
SubRepresentation	0 … N	specifies rovides information about a sub-representation that is embedded in the containing Representation. For more details see clause 8.3.4.5.
SegmentBase	0...1	specifies default Segment Base information. For more details see 8.4.4.
SegmentList	0 ... 1	specifies the Segment List information. For more details see 8.4.4.
SegmentTemplate	0 ... 1	specifies the Segment Template information. For more details see 8.4.4.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Elements are **bold**; attributes are non-bold and preceded with an @, List of elements and attributes is in ***italics bold*** referring to those taken from the base type that has been extended by this type.		

Table 8-14: XML-Syntax of `Representation` element

```
    <!-- A Representation of the presentation content for a specific Period -->
    <xs:complexType name="RepresentationType">
        <xs:complexContent>
            <xs:extension base="RepresentationBaseType">
                <xs:sequence>
                    <xs:element name="BaseURL" type="BaseURLType" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element name="SubRepresentation" type="SubRepresentationType"
minOccurs="0"/>
                    <xs:element name="SegmentBase" type="SegmentBaseType" minOccurs="0"/>
                    <xs:element name="SegmentList" type="SegmentListType" minOccurs="0"/>
                    <xs:element name="SegmentTemplate" type="SegmentTemplateType" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="id" type="StringNoWhitespaceType" use="required"/>
                <xs:attribute name="bandwidth" type="xs:unsignedInt" use="required"/>
                <xs:attribute name="qualityRanking" type="xs:unsignedInt"/>
                <xs:attribute name="mediaStreamStructureId" type="StringVectorType"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

  <!-- String without white spaces -->
  <xs:simpleType name="StringNoWhitespaceType">
    <xs:restriction base="xs:string">
      <xs:pattern value="[^\r\n\t \p{Z}]*"/>
    </xs:restriction>
  </xs:simpleType>

<!-- Whitespace-separated list of strings -->
<xs:simpleType name="StringVectorType">
  <xs:list itemType="xs:string"/>
</xs:simpleType>
```

8.4.3.5 Sub-Representation

Sub-Representations are embedded in regular Representations and are described by the **`SubRepresentation`** element. **`SubRepresentation`** elements are contained in a **`Representation`** element.

The **`SubRepresentation`** element describes properties of one or several media content components that are embedded in the Representation. It may for example describe the exact properties of an embedded audio component (language, codec, etc.), an embedded sub-title (language) or it may describe some embedded lower quality video layer (e.g. some lower frame rate, etc.).

Sub-Representations and Representation share some common attributes and elements.

In case the `@level` attribute is present in the **`SubRepresentation`** element,

- Sub-Representations provide the ability for accessing a lower quality version of the Representation in which they are contained. In this case, Sub-Representations for example allow extracting the audio track in a multiplexed Representation or may allow for efficient fast-forward or rewind operations if provided with lower frame rate.
- the Initialization Segment and/or the Media Segments shall provide sufficient information such that the data can be easily accessed through HTTP partial GET requests. The details on providing such information shall be defined by the media format in use. For media formats defined in this specification, the Subsegment Index as defined in section 9.2.3.3 shall be used.

If the `@level` attribute is absent, then the **`SubRepresentation`** element is solely used as a more detailed descriptor for media streams that are embedded in the Representation.

The semantics of the attributes and elements within a Sub-Representation are provided in Table 8-15. The XML-syntax of the Sub-Representation element is provided in Table 8-16.

Table 8-15: Semantics of `SubRepresentation` element

Element or Attribute Name	Use	Description
`SubRepresentation`		This element specifies a Sub-Representation.
`@level`	O	Specifies the sub-representation level. If `@level` attribute is present a Subsegment Index as defined in section 9.2.3.3 shall be available for each Media Segment in the containing Representation. If not present, then the **`SubRepresentation`** element is solely used to provide a more detailed description for media streams that are embedded in the Representation.
`@dependencyLevel`	O	specifies the set of Sub-Representations within this Representation that this Sub-Representation depends on in the decoding and/or presentation process as a whitespace-separated list of `@level` values. If not present, the Sub-Representation can be decoded and presented independently of any other Representation.
`@bandwidth`	O	Identical to the `@bandwidth` definition in Representation, but applied to this Sub-Representation. This attribute shall be present in case the `@level` attribute is present.
`@contentComponent`	O	if present, specifies the set of all media content components that are contained in this Sub-Representation as a whitespace-separated list of values of **`ContentComponent`**`@id` values. if not present, the Sub-Representation is not assigned to a media content component.
`CommonAttributesElements`	-	Common Adaptation Set, Representation and Sub-Representation attributes and elements (attributes and elements from base type ***RepresentationBaseType***), for details see clause 8.3.4.2.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) **Elements are** bold**;** attributes are non-bold and preceded with an @, List of elements and attributes is in ***italics bold*** referring to those taken from the base type that has been extended by this type.		

Table 8-16: XML-Syntax of `SubRepresentation` element

```
<!-- SubRepresentation of the presentation content for a specific Period;
    extends RepresentationBaseType -->
<xs:complexType name="SubRepresentationType">
    <xs:complexContent>
        <xs:extension base="RepresentationBaseType">
            <xs:attribute name="level" type="xs:unsignedInt"/>
            <xs:attribute name="dependencyLevel" type="UIntVectorType"/>
            <xs:attribute name="bandwidth" type="xs:unsignedInt"/>
            <xs:attribute name="contentComponent" type="StringVectorType"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<!-- Type for space delimited list of strings -->
<xs:simpleType name="UIntVectorType">
    <xs:list itemType="xs:unsignedInt"/>
</xs:simpleType>
```

8.4.3.6 Content Component

Each Adaptation Set contains one or more media content components. The properties of each media content component are described by a **`ContentComponent`** element or may be described directly on the **`AdaptationSet`** element if only one media content component is present in the Adaptation Set. **`ContentComponent`** elements are contained in an **`AdaptationSet`** element.

The semantics of the attributes and elements within a **ContentComponent** element are provided in Table 8-17. The XML syntax of the **ContentComponent** element is provided in Table 8-18.

Table 8-17 — Semantics of ContentComponent element

Element or Attribute Name	Use	Description
ContentComponent		description of a content component
@id	O	specifies an identifier for this media component. The attribute shall be unique in the scope of the containing Adaptation Set.
@lang	O	same semantics as in Table 8-11 for @lang attribute
@contentType	O	same semantics as in Table 8-11 for @contentType attribute
@par	O	same semantics as in Table 8-11 for @par attribute.
Accessibility	0 … N	same semantics as in Table 8-11 for **Accessibility** element
Role	0 … N	same semantics as in Table 8-11 for **Role** element
Rating	0 … N	same semantics as in Table 8-11 for **Rating** element
Viewpoint	0 … N	same semantics as in Table 8-11 for **ViewPoint** element

Legend:
For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory, F=Fixed.
For elements: <minOccurs>...<maxOccurs> (N=unbounded)
Elements are **bold**; attributes are non-bold and preceded with an @, List of elements and attributes is in ***italics bold*** referring to those taken from the base type that has been extended by this type.

Table 8-18 — XML-Syntax of ContentComponent element

```
<!-- Content Component -->
<xs:complexType name="ContentComponentType">
  <xs:sequence>
    <xs:element name="Accessibility" type="DescriptorType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="Role" type="DescriptorType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="Rating" type="DescriptorType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="Viewpoint" type="DescriptorType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:unsignedInt"/>
  <xs:attribute name="lang" type="xs:language"/>
  <xs:attribute name="contentType" type="xs:string"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

8.4.4 Segments and Segment Information

8.4.4.1 General

A Segment is the smallest addressable unit described by an MPD and has a defined format. Segment formats are defined in section 9. This clause defines the MPD information for Segments.

Specifically, a Segment shall referenced by an HTTP-URL included in the MPD, where an HTTP-URL is defined as an <absolute-URI> according to RFC 3986 [17], Clause 4.3, with a fixed scheme of 'http:' or "https: ", possibly restricted by a byte range if a range attribute is provided together with the URL. The byte range shall be expressed as a byte-range-spec as defined in RFC 2616 [9], Clause 14.35.1. It is restricted to a single expression identifying a contiguous range of bytes.

Each Segment referenced through an HTTP-URL in the MPD is associated with a Segment availability interval, i.e. a time window in wall-clock time at which the Segments can be accessed via the HTTP-URL. The Segment availability interval window is described by a Segment availability start time and a Segment availability end time.

Representations are assigned *Segment Information* through the presence of the elements **BaseURL**, **SegmentBase**, **SegmentTemplate** and/or **SegmentList**. The *Segment Information* provides information on the location, availability and properties of all Segments contained in one Representation. Specifically, information on the presence and location of Initialization, Media, Index and Bitstream Switching Segments is provided.

The elements **SegmentBase**, **SegmentTemplate** and **SegmentList** may be present in the **Representation** element itself. In addition, to express default values, they may be present in the **Period** and **AdaptationSet** element. At each level at most one of the three, **SegmentBase**, **SegmentTemplate** and **SegmentList** shall be present. Further, if **SegmentTemplate** or **SegmentList** on one level of the hierarchy, then the other one shall not be present on any lower hierarchy level.

SegmentBase, **SegmentTemplate** and **SegmentList** shall inherit attributes and elements from the same element on a higher level. If the same attribute or element is present on both levels, the one on the lower level shall take precedence over the one on the higher level.

Several mechanisms are available to specify the *Segment Information*. Specifically, each Representation shall have assigned exactly one of the following choices to determine the *Segment Information*, either by direct presence in the **Representation** element or by inheritance from the higher levels:

- one or more **SegmentList** elements - for syntax and semantics refer to section 8.4.4.2.3.
- one **SegmentTemplate** element - for syntax and semantics refer to section 8.4.4.2.4.
- one or more **BaseURL** elements, at most one **SegmentBase** element, and no **SegmentTemplate** or **SegmentList** element. The **SegmentBase** element is defined in section 8.4.4.2.2.

All three elements **SegmentBase**, **SegmentTemplate** and **SegmentList** share common elements based on the **SegmentBase** element. Furthermore, **SegmentTemplate** and **SegmentList** share common attributes and elements. The common information is defined in section 8.4.4.2.2.

The derivation and details of Initialization and Media Segment information based on the above information is provided in section 8.4.4.3.

8.4.4.2 Segment Information Description

8.4.4.2.1 Segment base information

The **SegmentBase** element contains information that is sufficient is only a single Media Segment is provided per Representation and the Media Segment URL is included in the **BaseURL** element.

In case multiple Media Segments are present, either a **SegmentList** or a **SegmentTemplate** is used that share the multiple Segment base information as provided in Table 8-20.

If the Representation contains more than one Media Segment, then the attribute `@duration` shall be present. Segments described by the Segment base information are referenced by an HTTP-URL conforming to the type `URLType` as defined in Table 8-21.

The semantics of the attributes and elements for the **SegmentBase** element and the Segment base information are provided in Table 8-19 and the multiple Segment base information in Table 8-20. The XML syntax of the Segment Base Information is provided in Table 8-22.

Table 8-19 — Semantics of `SegmentBase` element and *Segment Base Information* type

Element or Attribute Name	Use	Description
`SegmentBase` ***Segment Base Information***		specifies Segment base element. This element also specifies the type for the Segment base information that is the base type for other elements.
`@timescale`	O	specifies the timescale in units per seconds to be used for the derivation of different real-time

		duration values in the Segment Information. If not present on any level, it shall be set to 1. NOTE This may be any frequency but typically is the media clock frequency of one of the media streams (or a positive integer multiple thereof).
@presentationTimeOffset	O	specifies the presentation time offset of the Representation relative to the start of the Period. The value of the presentation time offset in seconds is the division of the value of this attribute and the value of the @timescale attribute. If not present on any level, the value of the presentation time offset is 0.
@timeShiftBufferDepth	O	specifies the duration of the time shifting buffer for this Representation that is guaranteed to be available for a Media Presentation with type 'dynamic'. When not present, the value of the @timeShiftBufferDepth on MPD level applies. If present, this value shall be not smaller than the value on MPD level. This value of the attribute is undefined if the type attribute is equal to "static". NOTE: When operating in a time-shift buffer on a Representation with value larger than the time-shift buffer signalled on MPD level, not all Representations may be available for switching.
@indexRange	O	specifies the byte range that contains the Segment Index in all Media Segments of the Representation. The byte range shall be expressed and formatted as a byte-range-spec as defined in RFC 2616 [9], Clause 14.35.1. It is restricted to a single expression identifying a contiguous range of bytes. If not present the value is unknown.
@indexRangeExact	OD default: "false"	when set to 'true' specifies that for all Segments in the Representation, the data outside the prefix defined by @indexRange contains the data needed to access all access units of all media streams syntactically and semantically. This attribute shall not be present if @indexRange is absent.
Initialization	0 ... 1	specifies the URL including a possible byte range for the Initialization Segment. For the type definition refer to Table 8-21.

Legend:

For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory.
For elements: <minOccurs>...<maxOccurs> (N=unbounded)
Elements are **bold**; attributes are non-bold and preceded with an @. List of elements and attributes is in ***italics bold*** referring to those taken from the base type that has been extended by this type.

Table 8-20 — Semantics of `MultipleSegmentBaseInformation` type

Element or Attribute Name	Use	Description
`MultipleSegmentBaseInformation`		specifies multiple Segment base information.
`@duration`	O	If present, specifies the constant approximate Segment duration. All Segments within this Representation element have the same duration unless it is the last Segment within the Period, which could be significantly shorter. The value of the duration in seconds is the division of the value of this attribute and the value of the `@timescale` attribute associated to the containing Representation. For more details refer to clause 8.4.4.4.3.
`@startNumber`	O	specifies the start number. The interpretation of the `@startNumber` depends on the segment addressing method. For more details refer to clause 8.4.4.4.3.
Segment Base Information		specifies Segment base information.

Legend:

For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory.
For elements: <minOccurs>...<maxOccurs> (N=unbounded)
Elements are **bold**; attributes are non-bold and preceded with an @. List of elements and attributes is in ***italics bold*** referring to those taken from the base type that has been extended by this type.

Table 8-21 — Semantics of elements of type `URLType`

Element or Attribute Name	Use	Description
Element of type **`URLType`**		defines an HTTP-URL
`@sourceURL`	O	specifies the source URL part and shall be formatted either as an `<absolute-URI>` according to RFC 3986, Clause 4.3, with a fixed scheme of 'http' or "https" or as a `<relative-ref>` according to RFC 3986, Clause 4.2. If not present, then any **`BaseURL`** element is mapped to the `@sourceURL` attribute and the range attribute shall be present.
`@range`	O	specifies the byte range restricting the above HTTP-URL. The byte range shall be expressed and formatted as a `byte-range-spec` as defined in RFC 2616, Clause 14.35.1. It is restricted to a single expression identifying a contiguous range of bytes. If not present, the element refers to the entire resource referenced in the `@sourceURL` attribute.

Legend:

For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory.
For elements: <minOccurs>...<maxOccurs> (N=unbounded)
Elements are **bold**; attributes are non-bold and preceded with an @. List of elements and attributes is in ***italics bold*** referring to those taken from the base type that has been extended by this type.

Table 8-22 — XML-Syntax of Segment Base Information

```
<!-- Segment information base -->
<xs:complexType name="SegmentBaseType">
  <xs:sequence>
    <xs:element name="Initialization" type="URLType" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="timescale" type="xs:unsignedInt"/>
  <xs:attribute name="presentationTimeOffset" type="xs:unsignedLong"/>
  <xs:attribute name="timeShiftBufferDepth" type="xs:duration"/>
  <xs:attribute name="indexRange" type="xs:string"/>
  <xs:attribute name="indexRangeExact" type="xs:boolean" default="false"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<!-- Multiple Segment information base -->
<xs:complexType name="MultipleSegmentBaseType">
  <xs:complexContent>
    <xs:extension base="SegmentBaseType">
      <xs:sequence>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="duration" type="xs:unsignedInt"/>
      <xs:attribute name="startNumber" type="xs:unsignedInt"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Segment Info item URL/range -->
<xs:complexType name="URLType">
  <xs:sequence>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="sourceURL" type="xs:anyURI"/>
  <xs:attribute name="range" type="xs:string"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

8.4.4.2.2 Segment list

The Segment list is defined by one or more **`SegmentList`** elements. Each **`SegmentList`** element itself contains a list of **`SegmentURL`** elements for a consecutive list of Segment URLs. Each Segment URL may contain the Media Segment URL and possibly a byte range. The Segment URL element may also contain an Index Segment.

The semantics of the attributes and elements for the Segment list are provided in Table 8-23. The XML syntax of the Segment list is provided in Table 8-24.

Table 8-23 — Semantics of `SegmentList` element

Element or Attribute Name	Use	Description
`SegmentList`		specifies Segment information.
`@xlink:href`	O	specifies a reference to a remote element that contains one or multiple elements of type **`SegmentList`**.
`@xlink:actuate`	OD default: "onRequest"	specifies the processing set, can be either "`onLoad`" or "`onRequest`"
`MultipleSegmentBaseInformation`		Multiple Segment base information as defined in Table 8-20.
`SegmentURL`	0 … N	specifies a Media Segment URL and a possibly present Index Segment URL

@media	O	in combination with the @mediaRange attribute specifies the HTTP-URL for the Media Segment. It shall be formatted as an <absolute-URI> according to RFC 3986, Clause 4.3, with a fixed scheme of 'http' or "https" or as a <relative-ref> according to RFC 3986, Clause 4.2. If not present, then any **BaseURL** element is mapped to the @media attribute and the range attribute shall be present.
@mediaRange	O	specifies the byte range within the resource identified by the @media corresponding to the Media Segment. The byte range shall be expressed and formatted as a byte-range-spec as defined in RFC 2616, Clause 14.35.1. It is restricted to a single expression identifying a contiguous range of bytes. If not present, the Media Segment is the entire resource referenced by the @media attribute.
@indexRange	O	specifies the byte range of the Segment Index in Media Segment. The byte range shall be expressed and formatted as a byte-range-spec as defined in RFC 2616, Clause 14.35.1. It is restricted to a single expression identifying a contiguous range of bytes. If not present, the Index Segment then no Index Segment information is provided for this Media Segment.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Note that the conditions only holds without using @xlink:href. If linking is used, then all attributes are "optional" and <minOccurs=0> Elements are **bold**; attributes are non-bold and preceded with an @. List of elements and attributes is in ***italics bold*** referring to those taken from the base type that has been extended by this type.		

Table 8-24 — XML-Syntax of SegmentList element

```
<!-- Segment List -->
<xs:complexType name="SegmentListType">
  <xs:complexContent>
    <xs:extension base="MultipleSegmentBaseType">
      <xs:sequence>
      <xs:element name="SegmentURL" type="SegmentURLType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute ref="xlink:href"/>
      <xs:attribute ref="xlink:actuate" default="onRequest"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Segment URL  -->
<xs:complexType name="SegmentURLType">
  <xs:sequence>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="media" type="xs:anyURI"/>
  <xs:attribute name="mediaRange" type="xs:string"/>
  <xs:attribute name="indexRange" type="xs:string"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
```

```
</xs:complexType>
```

8.4.4.2.3 Segment template

The Segment template is defined by the **`SegmentTemplate`** element. In this case, specific identifiers that are substituted by dynamic values assigned to Segments, to create a list of Segments. The substitution rules are provided in section 8.4.4.4.

The semantics of the attributes and elements for the Segment list are provided in Table 8-25. The XML syntax of the Segment Information is provided in Table 8-26.

Table 8-25 — Semantics of `SegmentTemplate` element

Element or Attribute Name	Use	Description
`SegmentTemplate`		specifies Segment template information.
MultipleSegmentBaseInformation		Provides the Multiple Segment base information as defined in Table 8-20.
`@media`	O	specifies the template to create the Media Segment List. For more details refer to clause 8.4.4.3.3.
`@initialization`	O	specifies the template to create the Initialization Segment. The `$Number$` identifier shall not be included. For more details refer to clause 8.4.4.3.2.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Elements are **bold**; attributes are non-bold and preceded with an @. List of elements and attributes is in ***italics bold*** referring to those taken from the base type that has been extended by this type.		

Table 8-26 — XML-Syntax of `SegmentTemplate` element

```
<!-- Segment Template -->
<xs:complexType name="SegmentTemplateType">
  <xs:complexContent>
    <xs:extension base="MultipleSegmentBaseType">
      <xs:attribute name="media" type="xs:string"/>
      <xs:attribute name="initialization" type="xs:string"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

8.4.4.3 Segment Information

8.4.4.3.1 Overview

The *Segment Information* provides the following information:

- the presence or absence of Initialization Segment information
- the HTTP-URL and possibly a byte range for each accessible Segment in each Representation,
- all valid Segment URLs declared by the containing MPD
- for services with **`MPD`**`@type='dynamic'`, the Segment availability start time and Segment availability end time of each Segment,,

- the approximate Media Presentation start time of a Media Segment in the Media Presentation timeline within the Period,

The derivation of Initialization and Media Segment Information is provided in subclause 8.4.4.3.2 and 8.4.4.3.3, respectively. Reference resolution as defined in section 8.7.2 and base URL selection as defined in section 8.7.3 using **`BaseURL`** elements as defined in section 8.7.1 shall be applied to any URLs.

8.4.4.3.2 Initialization Segment Information

Each Representation has assigned at most one Initialization Segment.

The presence of an Initialization Segment is indicated by the presence of **`SegmentBase.Initialization`**, **`SegmentList.Initialization`**, the **`SegmentTemplate.Initialization`** element or the **`SegmentTemplate`**`@initialization` attribute that may contain URL and byte range information or URL construction rules for the Initialization Segment.

If neither **`Initialization`** element nor **`SegmentTemplate`**`@initialization` are present for a Representation then each Media Segment within the Representation shall be self-initializing.

For services with **`MPD`**`@type='dynamic'` , the Segment availability start time of the Initialization Segment is the sum of the value of the **`MPD`**`@availabilityStartTime` and *PeriodStart* time and the Segment availability end time of the Initialization Segment is the largest Segment availability end time of any Media Segment in this Representation. For Segment availability for media Segments refer to clause 8.4.4.3.3.

The data structures retrieved from the Initialization URL are defined in section 9.2.2.

8.4.4.3.3 Media Segment Information

Each Representation has assigned a list of consecutive Media Segments. Each entry in the list of a media Segment has assigned the following parameters:

- a valid Media Segment URL and possibly a byte range.
- the number of the Media Segment in the Representation.
- the MPD start time of the Media Segment in the Representation providing an approximate presentation start time of the Segment
- MPD duration of the Media Segment providing an approximate presentation duration of the Segment

These parameters are specified by the **`SegmentTemplate`** or **`SegmentList`** elements. To obtain at least one entry in the list of Media Segments, one of the following shall apply:

- if **`SegmentTemplate`** element is present the Template-based Segment URL construction in section 8.4.4.4 shall be applied with the number of the Media Segment in the Media Segment list. The URL of the media segment at position *k* is determined by replacing the $*Number*$ identifier by (*k*-1) plus the value of the **`SegmentTemplate`**`@startNumber` attribute, if present, or is `1`.
- if one or more **`SegmentList`** elements are present they contain itself a list of **`SegmentURL`** elements for a consecutive list of Media Segment URLs. The first number in the list within this Period is determined by the value of the **`SegmentList`**`@startNumber` attribute, if present, or is 1 in case this attribute is not present. The sequence of multiple **`SegmentList`** elements within a Representation shall result in Media Segment List with consecutive numbers.
- none of the above: In this case only a single Media Segment shall be present with the URL provided by a **`BaseURL`** element and the **`SegmentBase`** element may be present.

The MPD start time is relative to the start of the Representation provided by the MPD. The MPD start time and the MPD duration are approximate and do not reflect the exact Media Presentation time. For more details on the relation of MPD start times and Media Presentation time refer to section 9.4.1.2.

For the derivation of the MPD start time and duration of each Media Segment, the *Index* of the Media Segment and the following information are used

- If the `@duration` attribute is not present, then the Representation shall contain exactly one Media Segment. The MPD start time is 0 and the MPD duration is obtained in the same way as for the last Media Segment in the Representation (see below for more details).

- If `@duration` attribute is present, then the MPD start time of the Media Segment is determined as (*Number--Number*$_{Start}$) times the value of the duration of the attribute `@duration` with *Number*$_{Start}$ the value of the `@startNumber` attribute. The MPD duration of the Media Segment is the value of the attribute `@duration` except for the duration of the last Media Segment (see below for more details).

- To determine the duration of the only or the last Media Segment of any Representation in a Period, the MPD shall include sufficient information to determine the duration of the containing Period. For example, the **`MPD`**`@mediaPresentationDuration`, or add **`Period`**`@duration`, or next **`Period`**`@start` may be present.

For services with **`MPD`**`@type='dynamic'`, the Segment availability start time of a Media Segment is the sum of the value of the **`MPD`**`@availabilityStartTime`, the *PeriodStart* time of the containing Period as defined in section 8.4.2, the MPD start time and the MPD duration of the Media Segment in the Representation. The Segment availability end time of a Media Segment is the sum of the Segment availability start time, the MPD duration of the Media Segment and the value of the attribute `@timeShiftBufferDepth` for this Representation.

NOTE: By adding the MPD duration of the segment to the segment availability start time of the segment, the segment availability start time of the first segment of each Period depends on the segment duration. This enables to provide segments in Representations with shorter MPD duration earlier, for example to reduce latency for certain Representations.

The MPD shall include URL information for all Segments with an availability start time less than both the end of the presentation and the sum of the latest time at which this version of the MPD is available on the server and the **`MPD`**`@minimumUpdatePeriod`.

The data structures retrieved from the URL referring to a Media Segment are defined in section 9.2.3.

8.4.4.4 Template-based Segment URL Construction

The **`SegmentTemplate`**`@media` attribute and the **`SegmentTemplate`**`@index` each contain a string that may contain one or more of the identifiers as listed in Table 8-27.

In each URL, the identifiers from Table 8-27 shall be replaced by the substitution parameter defined in Table 17. Identifier matching is case-sensitive. If the URL contains unescaped $ symbols which do not enclose a valid identifier then the result of URL formation is undefined. In this case it is expected that the DASH Client ignores the entire containing **`Representation`** element and the processing of the MPD continues as if this **`Representation`** element was not present. The format of the identifier is also specified in Table 8-27.

Each identfier may be suffixed, within the enclosing "$" characters, with an additional format tag aligned with the `printf` format tag as defined in IEEE 1003.1-2008 [s] following this prototype:

```
%0[width]d
```

The `width` parameter is an unsigned integer that provides the minimum number of characters to be printed. If the value to be printed is shorter than this number, the result shall be padded with zeros. The value is not truncated even if the result is larger.

The Media Presentation shall be authored such that the application of the substitution process results in valid Segment URLs.

Strings outside identifiers shall only contain characters that permit to form a valid HTTP-URL according to RFC 3986 [17].

Table 8-27: Identifiers for Segment Templates

$<Identifier>$	Substitution parameter	Format
$$	Is an escape sequence, i.e. "$$" is replaced with a single "$"	not applicable

$RepresentationID$	This identifier is substituted by the attribute **Representation**@id of the containing Representation.	The format tag shall not be present.
$Number$	This identifier is substituted by the *Number* of the corresponding Segment.	The format tag may be present. If no format tag is present, a default format tag with width=1 shall be used.

8.5 MPD Update

8.5.1 General

If the **MPD**@type is set to 'dynamic', the MPD may be updated during the Media Presentation. Updates typically extend the accessible Segment list for each Representation, introduce a new Period, update Segment locations or terminate the Media Presentation.

In this case the MPD shall be made accessible at all locations specified in any present **MPD.Location** element or, if none is present, at the same location as the initial MPD. If the client fetches the MPD using HTTP, the client should use conditional GET methods as specified in RFC 2616 [9], clause 9.3 to reduce unnecessary network usage in the downlink.

When the MPD is updated

- the value of **MPD**@id, if present, shall be the same in the original and the updated MPD;
- the values of any **Period**@id attributes shall be the same in the original and the updated MPD, unless the containing **Period** element has been removed.
- the values of any **AdaptationSet**@id attributes shall be the same in the original and the updated MPD unless the containing **Period** element has been removed.
- any Representation with the same @id and within the same Period as a Representation appearing in the previous MPD shall provide functionally equivalent attributes and elements, and shall provide functionally identical Segments with the same indices in the corresponding Representation in the new MPD.
- the value of **MPD**@availabilityStartTime shall be the same in the original and the updated MPD.

If the attribute **MPD**@minimumUpdatePeriod is not present, no update to the MPD is expected, the attribute **MPD**@mediaPresentationDuration shall be present and the MPD shall remain valid until the Media Presentation end time.

If the attribute **MPD**@minimumUpdatePeriod is present, updates to the MPD are expected and restricted in a sense that at the location where the MPD is available at a certain time, the MPD is also valid for the duration of the value of the **MPD**@minimumUpdatePeriod attribute. Specifically the following shall hold: .

If the *i*-th version of the MPD is the last version of MPD till the end of the Media Presentation, let *Texp(i)* be the Media Presentation end time. Otherwise, let *Texp(i)* be the sum of the value of **MPD**@minimumUpdatePeriod and the wall-clock time at which the *i*-th version of the MPD is updated (and replaced with the (*i*+1)-th version). The *i*-th MPD shall remain valid until *Texp*(*i*) in the following sense:

- all Segments with availability start time less than *Texp*(*i*) shall be available at their availability start times at the location advertised in the *i*-th MPD.
- all Representations have a Segment with an availability start time, *Tavail*, which is less than *Texp*(*i*) and with duration not less than (*Texp*(*i*) - *Tavail*). The actual duration of this Segment is not known by the client until this Segment or the next update of MPD is fetched and this duration may be less than the normal Segment duration if it is the last Segment of the Representation in this Period.

NOTE:

1) the actual duration of this Segment is not known at the client until this Segment or the updated MPD is fetched and this Segment duration may be less than the previous Segment duration if it is the last Segment in the Period.

2) The clients may not know *Texp*(*i*), but they can each calculate a lower bound on *Texp*(*i*) by adding **MPD**@minimumUpdatePeriod to the wall-clock time at which they request the MPD.

3) The second condition above ensures that sufficient media is contained in each Representation to present up to media presentation time *Texp*(*i*) for a client that begins playing each Segment at the earliest possible time (its availability start time).

4) The result of the MPD validity requirement is that all items a client expects to be able to retrieve (both Segments and MPD elements) are guaranteed to be available for retrieval during the periods that the client can expect them to be accessible.

5) An MPD may contain no Period element or only an early available Period may be provided. In this case, updates to the MPD are expected in order to provide the start time of the first Period, which coincides with the start of the actual Media Presentation.

8.5.2 Media Presentation Description Delta

If the **x3gpp:DeltaSupport** element is present in the **MPD** element, the content provider indicates that MPD delta files, as defined in this clause, are supported on the server. The URI of the MPD delta is provided in **x3gpp:DeltaSupport** @sourceURL. The **x3gpp:DeltaSupport** @availabilityDuration element, if present, indicates that the MPD delta file referenced by the URI is available for at least the value of the @availabilityDuration attribute (after this time, the server may redirect the client to the full MPD). If **x3gpp:DeltaSupport** @availabilityDuration is not present, then no information is conveyed about the availability of the MPD delta. If a client request for an MPD delta file results in an error, the client should request a full MPD.

The semantics of the attributes within the **x3gpp:DeltaSupport** element are provided in Table 8-28. The XML-syntax of **x3gpp:DeltaSupport** element is provided in Table 8-29.

Table 8-28: Semantics of x3gpp:DeltaSupport element

Element or Attribute Name	Use	Description
x3gpp:DeltaSupport		If present, this element indicates that MPD delta files are supported by the server.
@sourceURL	M	The source string providing the URL of the MPD delta. The URL may be relative to any **BaseURL** on MPD level and reference resolution according to clause 8.2.3 shall be applied.
@availabilityDuration	O	When provided, indicates the duration that the server guarantees the availability of the MPD delta file referenced in @sourceURL after the MPD has been updated. After that the client may be redirected to the full MPD.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Elements are **bold**; attributes are non-bold and preceded with an @.		

Table 8-29: XML-Syntax of x3gpp:DeltaSupport element

```
<!--DeltaSupport for the MPD -->
<xs:complexType name="DeltaSupportType">
    <xs:sequence>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="sourceURL" type="xs:anyURI" use="required"/>
    <xs:attribute name="availabilityDuration" type="xs:duration"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

An MPD delta is a text file that shall include the delta between the MPD that references it and the latest provided MPD. Note that the value of @sourceURL in successive MPDs is necessarily different because it is impossible for the delta between two different MPDs and the most recent MPD to be the same.

The output format consists of one or more structures, each corresponding to a change. The changes are in decreasing line number order. The structure format looks like:

```
change-command
to-file-line
to-file-line...
.
```

There are three types of change commands `change-command`. Each consists of a line number or comma-separated range of lines in the first file and a single character indicating the kind of change to make. All line numbers are the original line numbers in the file. The types of change commands and the instructions are provided in Table 8-30.

Table 8-30: Change commands and the instructions for delta MPD files

Change command	Instruction	Example
`la`	Add text from the second file after line `l` in the first file.	"`8a`" means to add the following lines after line 8 of file 1
`rc`	Replace the lines in range `r` in the first file with the following lines. Like a combined add and delete, but more compact.	"`5,7c`" means change lines 5–7 of file 1 to read as the text file 2.
`rd`	Delete the lines in range `r` from the first file.	"`5,7d`" means delete lines 5–7 of file 1.
NOTE: This is the format supported by the GNU diff utilities, see http://www.gnu.org/software/diffutils/manual/#Detailed-ed		

Regardless of the presence of a **`x3gpp:DeltaSupport`** element, the full MPD shall always be available to clients for regular MPD updates as defined in clause 8.5.1. MPD Delta related procedures are optional at the client.

8.6 Additional Media Presentation Information

8.6.1 Introduction

The MPD, Periods, Adaptation Sets, Representations and Sub-Representations may have assigned descriptors for describing the content or other elements in the MPD. This clause specifies this descriptive information.

8.6.2 Program Information

Descriptive information on the program may be provided for each period within the **`ProgramInformation`** element.

When multiple **`ProgramInformation`** elements are present, the `@lang` attribute shall be present and each element shall describe the Media Presentation sufficiently in the language defined by the value of the `@lang` attribute.

For each language, the program information may specify title, source of the program, copyright information and a URL to more information.

The semantics of the attributes within the **`ProgramInformation`** element are provided in Table 8-31. The XML-syntax of **`ProgramInformation`** element is provided in Table 8-32.

Table 8-31: Semantics of `ProgramInformation` element

Element or Attribute Name	Use	Description
`ProgramInformation`		specifies descriptive information about the program
`@lang`	O	Declares the language code(s) for this Program Information. The syntax and semantics according to IETF RFC 5646 [13] shall be applied. If not present the value is unknown.
`@moreInformationURL`	O	If specified, this attribute contains an absolute URL which provides more information about the Media Presentation in this Period.

		If not present the value is unknown.
Title	0...1	specifies a title for the Media Presentation
Source	0...1	specifies information about the original source (for example content provider) of the Media Presentation.
Copyright	0...1	specifies a copyright statement for the Media Presentation.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Elements are **bold**; attributes are non-bold and preceded with an @.		

Table 8-32: XML-Syntax of `ProgramInformation` element

```
<!-- Program information for a presentation -->
<xs:complexType name="ProgramInformationType">
    <xs:sequence>
        <xs:element name="Title" type="xs:string" minOccurs="0"/>
        <xs:element name="Source" type="xs:string" minOccurs="0"/>
        <xs:element name="Copyright" type="xs:string" minOccurs="0"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="lang" type="xs:language"/>
    <xs:attribute name="moreInformationURL" type="xs:anyURI"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

8.6.3 Descriptors

8.6.3.1 General

The MPD may contain descriptors that are all in the same format as defined in this clause. The elements of type `DescriptorType` provide a flexible mechanism for DASH content authors to annotate and extend the **`MPD`**, **`Period`**, **`AdaptationSet`** and **`Representation`** elements.

The descriptor elements are all structured in the same way, namely they contain a `@schemeIdUri` attribute to identify the scheme and an optional attribute `@value` and an optional attribute `@id`. The `@schemeIdUri` provides a URI to identify the scheme. The semantics of this element is specific to the scheme employed. The scheme may be a URN or a URL.

The MPD does not provide any specific information on how to use these elements. It is up to the application that employs DASH formats to instantiate the description elements with appropriate scheme information. Some specific schemes are defined in Annex C.

DASH applications that use one of these elements must first define a Scheme Identifier in the form of a URI and must then define the value space for the element when that Scheme Identifier is used. The Scheme Identifier appears in the `@schemeIdUri` attribute.

In the case that a simple set of enumerated values are required, a text string may be defined for each value and this string must be included in the `@value` attribute. If structured data is required then any extension element or attribute may be defined, but in a separate namespace.

The `@id` value may be used to refer to a unique descriptor or to a group of descriptors. In the latter case, descriptors with identical values for the attribute `@id` shall be synonymous, i.e. the processing of one of the descriptors with an identical value for `@id` is sufficient.

Two elements of type **`DescriptorType`** are *equivalent*, if the element name, the `@schemeIdUri` and the `@value` are equivalent. If the `@schemeIdUri` is a URN, then equivalence refers to lexical equivalence as defined in clause 5 of RFC 2141. If the `@schemeIdUri` is a URI, then equivalence refers to equality on a character-for-character basis as defined in clause 6.2.1 of RFC 3986 [17]. For the `@value` XML-string matching shall be used for determining equivalence. If the `@value` attribute is not present, equivalence is determined by the equivalence for `@schemeIdUri` only. The `@id` attribute may be ignored for equivalence determination.

The semantics of the attributes within a Generic Descriptor element are provided in Table 8-33. The XML-syntax of a Generic Descriptor element is provided in Table 8-34. The specific descriptors follow these syntax and semantics.

Table 8-33: Semantics of generic `Descriptor` element

Element or Attribute Name	Use	Description
Element of type `DescriptorType`		This element provides information about the use of description.
`@schemeIdUri`	M	Provides a URI to identify the scheme. The definition of this element is specific to the scheme employed for content description. The URI may be a URN or a URL. The `@schemeIdUri` may be a URN or URL. When a URL is used, it should also contain a month-date in the form mmyyyy; the assignment of the URL must have been authorized by the owner of the domain name in that URL on or very close to that date, to avoid problems when domain names change ownership
`@value`	O	This attribute provides the value for the descriptor element. The value space and semantics must be defined by the owners of the scheme identified in the `@schemeIdUri` attribute.
`@id`	O	specifies an identifier for the descriptor. Descriptors with identical values for this attribute shall be synonymous, i.e. the processing of one of the descriptors with an identical value is sufficient.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Elements are **bold**; attributes are non-bold and preceded with an @.		

Table 8-34: XML-Syntax of generic `Descriptor` element

```
<!-- Generic named descriptive information -->
<xs:complexType name="DescriptorType">
    <xs:sequence>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="schemeIdUri" type="xs:anyURI" use="required"/>
    <xs:attribute name="value" type="xs:string" use="optional"/>
    <xs:attribute name="id" type="xs:string"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

8.6.3.2 Content Protection

For the element **`ContentProtection`** the `@schemeIdUri` attribute is used to identify the content protection schemes employed. This attribute should provide sufficient information, possibly in conjunction with the `@value` and/or extension attributes and elements, such as the DRM system(s), encryption algorithm(s), and key distribution scheme(s) employed, to enable a client to determine whether it can possibly play the protected content. The **`ContentProtection`** element can be extended in a separate namespace to provide information specific to the content protection scheme (e.g. particular key management systems or encryption methods). Scheme-specific information can also be provided in the Initialization Segment(s) using the appropriate file format primitives instead of, or in addition to, the **`ContentProtection`** element. The client may have to receive and analyze the protected content (typically only the Initialization Segment, if present), before it can determine whether it has already acquired a license and/or key for accessing the protected content, or to determine from where it can acquire a missing license and/or key, in case this information is not available from the **`ContentProtection`** element.

When the **`ContentProtection`** element is not present the content shall neither be encrypted nor content protected.

When multiple **`ContentProtection`** elements are present, each element shall describe a content protection scheme that is sufficient to access and present the Representation.

8.6.3.3 Role

For the element **Role** the @schemeIdUri attribute is used to identify the role scheme employed to identify the role of the media component. Roles define and describe characteristics and/or structural functions of media components.

One Adaptation Set or one media content component may have assigned multiple roles even within the same scheme.

This specification defines a role scheme in Annex C.2.

8.6.3.4 Rating

For the element **Rating** the @schemeIdUri attribute is used to identify the rating scheme employed.

Ratings specifies that content is suitable for presentation to audiences for which that rating is known to be appropriate, or for unrestricted audiences.

NOTE: An audience with a rating restriction is intended to not be presented content that has associated ratings, unless at least one scheme is recognized as indicating that the content is appropriate to that audience.

8.6.3.5 Viewpoint

For the element **Viewpoint** the @schemeIdUri attribute is used to identify the viewpoint scheme employed.

Adaptation Sets containing non-equivalent **Viewpoint** contain different media content components. The **Viewpoint** elements may equally be applied to media content types that are not video. .

Adaptation Sets with equivalent **Viewpoint** element values are intended to be presented together. This handling should be applied equally for recognised and unrecognised @schemeIdUri values.

8.6.3.6 Accessibility

For the element **Accessibility** the @schemeIdUri attribute is used to identify the accessibility scheme employed. Accessibility is a general term used to describe the degree to which the DASH Media Presentation is available to as many people as possible.

NOTE **Accessibility** elements fulfil a very similar purpose with respect to media content components as for **Role** elements, but are specifically intended for accessibility.

One Adaptation Set or one media content component may have assigned multiple accessibility purposes even within the same scheme.

This specification does not define a specific accessibility scheme, but the simple role scheme in may be used to express a minimum amount of accessibility information.

8.6.3.7 Audio channel configuration

For the element **AudioChannelConfiguration** the @schemeIdUri attribute is used to identify the audio channel configuration scheme employed.

Multiple **AudioChannelConfiguration** elements may be present indicating that the Representation supports multiple audio channel configurations. For example, it may describe a Representation that includes MPEG Surround audio supporting stereo and multichannel.

NOTE if the scheme or the value for this descriptor is not recognized the DASH client is expected to ignore the descriptor.

8.6.3.8 Essential Property Descriptor

For the element **EssentialProperty** the Media Presentation author expresses that the successful processing of the descriptor is essential to properly use the information in the parent element that contains this descriptor unless the element shares the same @id with another **EssentialProperty** element.

If **EssentialProperty** elements share the same @id, then processing one of the **EssentialProperty** elements with the same value for @id is sufficient. At least one **EssentialProperty** element of each distinct @id value is expected to be processed.

NOTE if the scheme or the value for this descriptor is not recognized the DASH client is expected to ignore the parent element that contains the descriptor.

Multiple **EssentialProperty** elements with the same value for @id and with different values for @id may be present.

8.6.3.9 Supplemental Property Descriptor

For the element **SupplementalProperty** the Media Presentation author expresses that the descriptor contains supplemental information that may be used by the DASH client for optimized processing.

NOTE if the scheme or the value for this descriptor is not recognized the DASH client is expected to ignore the descriptor.

Multiple **SupplementalProperty** elements may be present.

8.7 Base URL Processing

8.7.1 General

The **BaseURL** element may be used to specify one or more common locations for Segments and other resources. Reference resolution as defined in 8.7.2 shall be applied to each URL in the MPD. Handling of multiple alternative base URLs is addressed in 8.7.3.

The semantics of the attributes and elements for the Base URL are provided in Table 8-35. The XML syntax of the Base URL is provided in Table 8-36.

Table 8-35 — Semantics of BaseURL element

Element or Attribute Name	Use	Description
BaseURL		A URL that can be used as Base URL. The content of this element is a URI string as described in 8.7.2.
@serviceLocation	O	This attribute specifies a relationship between BaseURLs such that **BaseURL** elements with the same @serviceLocation value are likely to have their URLs resolve to services at a common network location, for example a common Content Delivery Network. If not present, no relationship to any other Base URL is known.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Elements are **bold**; attributes are non-bold and preceded with an @.		

Table 8-36 — XML-Syntax of BaseURL element

```
<!-- Base URL -->
```

```
<xs:complexType name="BaseURLType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="serviceLocation" type="xs:string"/>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

8.7.2 Reference resolution

URLs at each level of the MPD are resolved according to RFC3986 with respect to the **BaseURL** element specified at that level of the document or the level above in the case of resolving base URLs themselves (the document 'base URI' as defined in RFC 3986 [17], Section 5.1 is considered to be the level above the MPD level). If only relative URLs are specified and the document base URI cannot be established according to RFC3986 [17] then the MPD should not be interpreted. URL resolution applies to all URLs found in MPD documents.

In addition to the document level (the level above the MPD level), base URL information may be present on the following levels:

- On MPD level in **MPD.BaseURL** element. For details refer to section 8.4.1.
- On Period level in **Period.BaseURL** element. For details refer to section 8.4.2.
- On Adaptation Set level in **AdaptationSet.BaseURL** element. For details refer to section 8.4.3.3.
- On Representation level in **Representation.BaseURL**. For details refer to section 8.4.3.4.

8.7.3 Alternative base URLs

If alternative base URLs are provided through the **BaseURL** element at any level, identical Segments shall be accessible at multiple locations. In the absence of other criteria, the DASH Client may use the first **BaseURL** element as 'base URI''. The DASH Client may use base URLs provided in the **BaseURL** element as 'base URI' and may implement any suitable algorithm to determine which URLs it uses for requests.

9 DASH - Usage of 3GPP File Format

9.1 Introduction

3GPP Dynamic Adaptive Streaming over HTTP uses many elements of fragmented 3GP files to define the Segment formats. This provides Segments according to the requirements defined in clause 8.4.4.1 and enables reuse of existing content, easy encoding and recording, etc. This clause introduces how to use the 3GPP file format as specified in TS 26.244 [4] for DASH Segment formats.

9.2 Segment Types and Formats

9.2.1 Introduction

3GP-DASH defines a Segment format that is used in the delivery of media data over HTTP. A Segment shall contain one or more boxes in accordance with the boxed structure of the ISO-base media file format [11].

For 3GP-DASH the following applies:

- In all cases for which a Representation contains more than one Media Segment, the following applies:

  - The Initialization Segment as defined in clause 9.2.2 shall be present. The Initialization Segment shall be available for the 3GP-DASH client before any Media Segment is processed within the Representation.

  - Media Segments shall not be self-initializing. The Media Segment format is defined in clause 9.2.3.

- In case a Representation contains only a single Media Segment, then either one of the following two options is used:

  1) An Initialization Segment as defined in clause 9.2.2 and one Media Segment as defined in clause 9.2.3.

  2) One Self-Initializing Media Segment as defined in clause 9.2.4.

9.2.2 Initialization Segment

The Initialization Segment is conformant with the 3GPP file format, adaptive streaming profile and shall carry '3gh9' as compatibility brand.

The Initialization Segment consists of the 'ftyp' box, the 'moov' box, and optionally the 'pdin' box. The 'moov' box shall not contain any samples (i.e. the entry_count in the 'stts', 'stsc', and 'stco' boxes shall be set to 0) and is then very small in size. This reduces the start-up time significantly as the Initialization Segment needs to be downloaded before any Media Segment can be processed.

The 'mvex' box shall be contained in the 'moov' box to indicate that the client has to expect movie fragments. The 'mvex' box also sets default values for the tracks and samples of the following movie fragments.

The Initialization Segment provides the client with the metadata that describes the media content. The client uses the information in the 'moov' box to identify the available media components and their characteristics.

The Initialization Segment shall not contain any 'moof' or 'mdat' boxes.

9.2.3 Media Segment

9.2.3.1 General

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.

In addition, a Media Segment

1. shall contain a number of complete access units.
2. should contain at least one Stream access point (SAP) for each contained media stream.
3. should provide information on how to access the Media Presentation within this Segment, e.g. exact presentation time and an index. There is no requirement that a Media Segment starts with a SAP, but it is possible to signal in the MPD that all media streams in a Segments within a Representation start with a SAP.
4. if it is the first Media Segment in the Representation, it shall contain only media streams that start with a SAP of type 1 or 2.
5. shall contain sufficient information to time-accurately present each contained media component in the Representation without accessing any previous Media Segment in this Representation provided that the Media Segment contains a SAP for each media stream. The time-accuracy enables a client to seamlessly switch Representations and jointly present multiple Representations.
6. may be divided into Subsegments by a Segment Index as defined in 9.2.3.2.
7. shall specify all Media Presentation times relative to the start of the Period and compensated with the value of the @presentationTimeOffset. The presentation time in Media Segments shall be accurate to ensure accurate alignment of all Representations in one Period. For more details refer to 9.4.1.1.

9.2.3.2 Subsegments and Segment Index

Media Segments may contain multiple Subsegments. Each Subsegment shall contain a number of complete access units. There may also be media-format-specific restrictions on Subsegment boundaries. If a Segment is divided into multiple Subsegments this division is described by a compact Segment index, which provides the presentation time range in the

Representation and corresponding byte range in the Segment occupied by each Subsegment for one or more media streams. Clients may download this index in advance and then issue requests for individual Subsegments.

In addition, the Segment Index provides timing and stream access information. This includes the earliest presentation time of access units in each Subsegment of an indexed media stream and the presentation time of the first SAP, if present.

If a Segment Index is present for at least one media stream, then for any media stream for which no Segment Index is present, referred to as non-indexed stream, the following applies:

- every access unit of the non-indexed streams shall be a SAP of type 1.
- for each Subsegment, every non-indexed stream must contain exactly one access unit within the Subsegment with presentation time less than or equal to the earliest presentation time of the Subsegment

When multiple media streams are indexed in a single index file, the corresponding Segment Index for different media streams should index the same number of Subsegments.

If no Segment Index is provided for a Media Segment, then the Media Segment constitutes one Subsegment.

A Subsegment may itself be further subdivided using further Segment Index boxes. If a Subsegment only contains media data but no Segment Index, it is referred to as Media Subsegment.

1) The Segment Index may contain additional Subsegment indexing information for accessing different levels of Subsegments in a Media Subsegment. For more details refer to clause 9.2.3.3.

A generic mechanism for indexing of Media Segments is provided by the Segment Index ("`sidx`") box in TS26.244 [4]. In this case,

- the earliest presentation time of a Subsegment is documented in the `earliest_presentation_time` field.
- the byte range is document by the `first_offset` field and the `reference_size` field. If two Segment Index boxes document the same byte range, then the value of their `first_offset` field and their `reference_size` field shall be identical.

9.2.3.3 Subsegment Index

Media Subsegments may be indexed further to enable accessing different levels of Subsegments in a Media Subsegment. This Subsegment Index may also be provided in separate Index Segments together with the Segment Index.

A generic syntax and semantic for Subsegment indexing is provided by the Subsegment Index ("`ssix`") in Annex G.3.

9.2.3.4 3GP-DASH Media Segment Format

A Media Segment conforming to the Media Segment Format for 3GP DASH shall carry "`3gmA`" as a compatible brand and is defined as follows:

- Each Media Segment may contain an "`styp`" box.
- If the Media Segment is the last media Segment in the Representation, the `'styp'` box may carry "`lmsg`" as a compatible brand.
- Each Media Segment shall contain one or more whole self-contained movie fragments. A whole, self-contained movie fragment is a movie fragment ("`moof`") box and a media data ("`mdat`") box that contains all the media samples that do not use external data references referenced by the track runs in the movie fragment box.
- Each "`moof`" box shall contain at least one track fragment.
- The "`moof`" boxes shall use movie-fragment relative addressing and the flag "`default-base-is-moof`" shall also be set. Absolute byte-offsets shall not be used. In a movie fragment, the durations by which each track

extends should be as close to equal as practical. In particular, as movie fragments are accumulated, the track durations should remain close to each other and there should be no 'drift'.

- Each "traf" box shall contain a "tfdt" box.
- The track fragment adjustment box "tfad" as defined in 3GPP TS26.244 [4] may also be present to maintain compatibility with earlier releases of this specification; care should be taken that the alignment established by the "tfdt" and the time-shifting implied by the "tfad" not be both applied, which would result in a double correction.
- Each Media Segment may contain one or more "sidx" boxes. If present, the first "sidx" box shall be placed before any "moof" box and the subsegment documented by the first Segment Index ("sidx") box shall be the entire Segment, i.e. the entire Segment shall be document by the first Segment Index ("sidx") box.
- A media Segment may contain a Subsegment Index box ("ssix"). If present it shall follow immediately after the "sidx" box that documents the same subsegment. This immediately preceding "sidx" shall only index subsegments.
- Further rules on media Segments in combination with certain MPD attributes are provided in clause 9.4.

9.2.4 Self-Initializing Media Segment

A Self-Initializing Media Segment conforms to the concatenation of an Initialization Segment as defined in 9.2.2 and a Media Segment as defined in 9.2.3.

9.2.5 Media Stream and Segment Properties

9.2.5.1 Media stream access points

To be able to access a Representation, each of the media streams that are contained in the Representation requires Media Stream Access Points (SAPs). Annex G.6 defines different types of SAPs that provide a relationship between the position where a stream can be accessed, a SAP, relative to the start of a Segment or Subsegment, its presentation time and the presentation times and position of other access unit in the stream.

A SAP is a position in a Representation that enables 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).

For each SAP the properties, I_{SAP}, T_{SAP}, I_{SAU}, T_{DEC}, T_{EPT}, and T_{PTF} are identified and defined in Annex G.6.2.

In particular, T_{SAP} is defined to be earliest presentation time of any access unit of the media stream such that all access units of the media stream with presentation time greater than or equal to T_{SAP} can be correctly decoded using data in the Representation starting at byte position I_{SAP} and no data before I_{SAP}.

9.2.5.2 Non-overlapping Segments and Subsegments

Segments and Subsegments represent units for which the client has an exact map on how to access and download the unit using HTTP GET or HTTP partial GET methods.

Segments (respectively Subsegments) are typically generated by segmenting encoded media streams into appropriate units. If the generation of Segments (respectively Subsegments) adheres to certain rules, then the sequential decoding and presentation of Media Segments (respectively Subsegments) results in a correct presentation of all contained media streams. To define such rules the notion of 'non-overlapping' Segments (respectively Subsegments) is defined as follows.

Let

- $T_E(S,i)$ be the earliest presentation time of any access unit in stream *i* of a Segment or Subsegment *S*,
- $T_L(S,i)$ be the latest presentation time of any access unit in stream *i* of a Segment or Subsegment *S*.

Then two Segments (respectively Subsegments), *A* and *B*, which may or may not be of different Representations, are *non-overlapping* if $T_L(A,i) < T_E(B,i)$ for all media streams *i* in A and B or if $T_L(B,i) < T_E(A,i)$ for all streams *i* in A and B where *i* refers to the same media component.

The property of 'non-overlapping' Segments (respectively Subsegments) is used to define the terms Segment alignment and Subsegment alignment.

9.2.5.3 Bitstream concatenation

A sequence of Segments (respectively Subsegments) is a "conforming Segment (respectively Subsegment) sequence" if the concatenation of all Segments (respectively Subsegments) in the sequence of Segments (respectively Subsegments) results in a bitstream that conforms to the media formats in use (including container and codecs).

NOTE This implies that a player conforming to the media format can play the resulting bitstream.

9.3 Usage on Server and Client

3GP-DASH uses 3GP files according to the 3GP Adaptive-Streaming profile as specified in TS 26.244 [4]. Content may be prepared as 3GP files according to the 3GP Adaptive-Streaming profile. Initialization Segments and Media Segments may be generated by segmenting such 3GP files. Segment Index "`sidx`" boxes may be pre-contained in 3GP files or may be generated during the segmentation process. Clients may store a concatenation of a received Initialization Segment and a sequence of Media Segments from the same Representation to create a compliant 3GP file according to the Adaptive Streaming profile without accessing any media samples.

NOTE: As specified in TS 26.244, the MPD may be linked or embedded in the "`meta`" box of the "`moov`" box. This enables clients to access the MPD from a 3GP file that was made available from other means than 3GP-DASH (e.g. progressive download).

9.4 Segment Properties with MPD constraints

9.4.1 General

9.4.1.1 Introduction

The content, especially the Segments across Representations at the same media time may have been prepared in a joint or at least coordinated manner. To expose these properties to the client, certain flags in the MPD can be set to true to indicate such coordinated content preparation. Clients consuming 3GP-DASH formatted media presentations may benefit from properly authored content when switching between or presenting Representations.

9.4.1.2 Media Presentation Timeline

One of the key features in DASH is that encoded versions of different media components share a common timeline. The presentation time of access unit within the media content is mapped to the global common presentation timeline for synchronization of different media components and to enable seamless switching of different coded versions of the same media components.

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. This means for an access unit with a presentation time T_P signalled in the media stream, the Media Presentation time relative to the *PeriodStart* is $T_M=T_P - T_O$.

Media Segments should not contain any presentation time T_P that is smaller than the value of the `@presentationTimeOffset`, T_O. However, if this is the case, then presentation of the Media Segment is expected to only take place for presentation times greater than or equal to T_O.

The MPD start times as defined in 8.4.4.3.3 shall provide an approximation of the Media Presentation time T_M within the Period. Specifically, the MPD start time shall be drift-free relative to the presentation time T_P signalled in the media stream, i.e. the accuracy of the offset of the MPD start time relative to the presentation time does not depend on the position of the Segment in the Representation.

NOTE At the start of a new Period, the playout procedure of the media content components may need to be adjusted at the end of the preceding Period to match the *PeriodStart* time of the new Period as there may be small overlaps or gaps with a Representation at the end of the preceding Period. Overlaps (respectively gaps) may result from Media Segments with actual presentation duration of the media stream longer (respectively shorter) than indicated by the Period duration. Also in the beginning of a Period if the earliest presentation time T_P of any access unit of a Representation is larger than 0 then the playout procedures need to be adjusted accordingly.

For the case when **MPD**@type is "dynamic" and the attribute **MPD**@suggestedPresentationDelay is present, then the sum of value of the the **MPD**@availabilityStartTime, the *PeriodStart* value, the presentation time within the Period of an access unit, T_M, and the value of the the attribute **MPD**@suggestedPresentationDelay provides a mapping of the presentation time of each access unit to the wall-clock time, for example to express synchronization with a content internal time or for other reasons to enable synchronization of presentation to the wall-clock.

For the Segment formats as defined in section 9.2.3.4, the presentation time T_P internal in the media that maps the media to the Media Presentation timeline shall be relative to the movie timeline, i.e. they are composition times after the application of any edit list for the track.

It is recommended that the @timescale attribute in the MPD matches the timescale field in the Media Header Box of a present track. If the Segment Index ('sidx') box is present, then it is further recommended that the track for which the Segment Index ('sidx') box that appears first in the Media Segment is the track defining the value of the @timescale attribute.

9.4.1.3 Segment Index

If a Segment Index is present in a Media Segment of one Representation within an Adaptation Set, then the following shall hold:

- the order of Segment Index boxes for multiple media streams induces an ordering on the media content components equal to the order in which a Segment Index box for a media stream for each component first appears. This ordering shall be the same for all Segments of all Representations of an Adaptation Set. As a consequence, if there is a Segment Index for a media content component in one Segment there shall be a Segment Index for that media component in all Segments in this Adaptation Set.
- non-indexed media streams in all Representations of an Adaptation Set shall have the same access unit duration.

9.4.2 Segment Alignment

No additional requirements beyond those stated in section 8.4.3.3 are defined.

9.4.3 Bitstream Switching

If the @bitstreamSwitching is set to "true" for a set of Representations within an Adaptation Set, the conditions stated in section 8.4.3.3 shall be satisfied.

As a consequence of @bitstreamSwitching being set to "true", the following conditions are satisfied:

- The track IDs for the same media content component are identical for each Representation in each Adaptation Set.
- The conditions required for setting the @segmentAlignment attribute to a value other than 'false' for the Adaptation Set are fulfilled.

- The conditions required for setting (i) the `@startWithSAP` attribute to 2 for the Adaptation Set, or (ii) the conditions required for all Representations within the Adaptation Set to share the same value of `@mediaStreamStructureId` and setting the `@startWithSAP` attribute to 3 for the Adaptation Set, are fulfilled.

9.4.4 Sub-Representation

If a **`SubRepresentation`** element is present in a Representation in the MPD and the **`SubRepresentation`**`@level` is present, then the media Segments in this Representation shall include a Segment Index ("`sidx`") box and the Initialization Segment shall contain the Level Assignment ("`leva`") box.

The attribute `@level` specifies the level to which the described Sub-Representation is associated in the Subsegment Index. Level *n* corresponds to the *n*-th level in the Subsegment Index. The information in Representation, Sub-Representation and in the Level Assignment ("`leva`") box contains information on the assignment of media data to levels.

Media data should be ordered such that each higher value for `@level` provides an enhancement compared to any lower value of `@level`.

For temporal level assignment, the sample grouping `'tele'` as defined in clause G.4, shall be used.

10 QoE for Progressive Download and DASH

10.1 General

A progressive download or 3GP-DASH client supporting Quality of Experience (QoE) shall report QoE metrics according to the QoE configuration. QoE reporting is optional, but if a 3GP-DASH client reports DASH metrics, it shall report all requested metrics.

The quality metrics are defined in subclause 10.2.

The quality metrics applicable for progressive download are specified in section 10.3. In this case the activation and configuration of QoE reporting framework is achieved by a corresponding OMA DM QoE Management Object as specified in Annex F.

The quality metrics for DASH are specified in section 10.4. In this case, QoE reporting may be triggered using the MPD (i.e. when the **`Metrics`** element is present in the MPD) or using OMA DM QoE Management Object as specified in Annex F. When QoE reporting is triggered via the MPD or OMA DM QoE Management Object, the 3GP-DASH client is expected to collect quality metrics according to the QoE configuration. When using the MPD, the Quality Reporting scheme as defined in section 10.5 may be used.

The quality metric reporting protocol is defined in subclause 10.6. This protocol shall be used when QoE reporting is triggered via the MPD or OMA DM QoE Management Object.

10.2 QoE Metric Definitions

10.2.1 Introduction

This section provides the general QoE metric definitions and measurement framework.

The semantics are defined using an abstract syntax. Section 10.6 provides a mapping to an XML schema. Items in this abstract syntax have one of the following primitive types (`Integer`, `Real`, `Boolean`, `Enum`, `String`) or one of the following compound types:

- `Objects`: an unordered sequence of (`key`, `value`) pairs, where the key always has string type and is unique within the sequence.

- `List`: a ordered list of items.

- `Set`: an unordered set of items.

Additionally, there are two kinds of timestamp defined, i.e. *real time* (wall-clock time) and *media time*.

10.2.2 HTTP Request/Response Transactions

Table 25 contains the metric defining the List of HTTP Request/Response Transactions.

Table 25: List of HTTP Request/Response Transactions

Key	Type	Description
`HttpList`	`List`	List of HTTP request/response transactions
`Entry`	`Object`	An entry for a single HTTP request/response
`tcpid`	`Integer`	Identifier of the TCP connection on which the HTTP request was sent.
`type`	`Enum`	This is an optional parameter and should not be included in HTTP request/response transactions for progressive download. The type of the request: - MPD - MPD delta file - XLink expansion - Initialization Segment - Index Segment - Media Segment
`url`	`String`	The original URL (before any redirects or failures)
`actualurl`	`String`	The actual URL requested, if different from above
`range`	`String`	The contents of the `byte-range-spec` part of the HTTP Range header.
`trequest`	`Real Time`	The real time at which the request was sent.
`tresponse`	`Real Time`	The real time at which the first byte of the response was received.
`responsecode`	`Integer`	The HTTP response code.
`interval`	`Integer`	The duration of the throughput trace intervals (ms), for successful requests only.
`Trace`	`List`	Throughput trace, for successful requests only.
`Entry`	`Object`	A single throughput measurement entry.
`s`	`Real Time`	Measurement period start.
`d`	`Integer`	Measurement period duration (ms).
`b`	`List`	List of integers counting the bytes received in each trace interval within the measurement period.

NOTE:

1) Information additional to that specified in the `type` may be returned, for example if a client makes a request for a initialization information from a self-initializing Media Segment then index information may also be received.

2) All entries for a given object will have the same `url` and `range` and so can easily be correlated. If there were redirects or failures there will be one entry for each redirect/failure. The redirect-to URL or alternative URL (where multiple have been provided in the MPD) will appear as the `actualurl` of the next entry with the same `url` value.

3) The periods reported in `Entry` should be those periods where the client was actively reading from the TCP connections (i.e. they should not include periods where the TCP connection is idle due to zero receive window).

The end of the last measurement period reported in the `Trace` shall be the time at which the last byte of the response was received.

The `interval` and `Trace` shall be absent for redirect and failure records.

The key HttpList(*n*,*type*) where *n* is a positive integer is defined for an HttpList with an interval of *n* ms and *type* is one of MPD, MPDDeltaFile, XLinkExpansion, InitializationSegment, MediaSegment, or IndexSegment. If *type* is not present, all HTTP transactions are requested to be collected. If *type* is present, it specifies that the HTTP transactions concerning a resource equal to *type* are requested to be collected. Multiple keys HttpList(*n*,*type*) with different values of *n* and *type* may be present for a single @metrics attribute value.

An HTTP transaction that is not finished within a QoE metric collection period shall not be included in the reported metrics.

10.2.3 Representation Switch Events

Table 26 defines the metric to report a list of representation switch events.

Table 26: List of Representation Switch Events

Key			Type	Description
RepSwitchList			List	List of representation switch events (a switch event is the time at which the first HTTP request for a new representation, that is later presented, is sent)
	Entry		Object	A representation switch event.
		t	Real Time	Time of the switch event.
		mt	Media Time	The media time of the earliest media sample (out of all media components) played out from the 'to' representation.
		to	String	Value of Representation@id identifying the switch-to representation.
		Lto	Integer	If present, value of **SubRepresentation**@level within Representation identifying the switch-to level of the Representation

10.2.4 Average Throughput

This metric in Table 27 indicates the average throughput that is observed by the client during the measurement interval.

Table 27: Average Throughput

Key		Type	Description
AvgThroughput		Object	Average throughput that is observed by the client during the measurement interval
	numbytes	Integer	The total number of the content bytes, i.e. the total number of bytes in the body of the HTTP responses, received during the measurement interval.
	activitytime	Integer	The activity time during the measurement interval in milliseconds. The activity time during the measurement interval is the time during which at least one GET request is still not completed (i.e. excluding inactivity time during the measurement interval).
	t	Real Time	The real time of the start of the measurement interval
	duration	Integer	The time in milliseconds of the measurement interval
	accessbearer	String	Access bearer for the TCP connection for which the average throughput is reported
	inactivitytype	Enum	Type of the inactivity, if known and consistent throughout the reporting period: User request (e.g. pause) Client measure to control the buffer Error case

If the client requests the media Segments from the server separately over multiple non-competing parallel TCP connections established over separate access network bearers named as accessbearer, then the average throughput

values should be reported as a list of events with average throughput for each access network and associated access network bearer information reported separately, following the same guidelines as described above.

10.2.5 Initial Playout Delay

This metric in Table 28 signals the initial playout delay at the start of the streaming of the presentation.

Table 28: Initial Playout Delay

Key	Type	Description
`InitialPlayoutDelay`	`Integer`	The initial playout delay is measured as the time in milliseconds from the fetch of the first media Segment (or sub-segment) and the time at which media is retrieved from the client buffer.

10.2.6 Buffer Level

Table 29 defines the metric to report a list of buffer level status events.

Table 29: List of Buffer Level

Key			Type	Description
`BufferLevel`			`List`	List of buffer occupancy level measurements during playout at normal speed.
	`Entry`		`Object`	One buffer level measurement.
		`t`	`Real Time`	Time of the measurement of the buffer level.
		`level`	`Integer`	Level of the buffer in milliseconds. Indicates the playout duration for which media data of all active media components is available starting from the current playout time.

The key is `BufferLevel(`*`n`*`)`, where *n* is a positive integer is defined to refer to the metric in which the buffer level is recorded every *n* ms.

10.2.7 Play List

Decoded samples are generally rendered in presentation time sequence, each at or close to its specified presentation time. A compact representation of the information flow can thus be constructed from a list of time periods during which samples of a single representation were continuously rendered, such that each was presented at its specified presentation time to some specific level of accuracy (e.g. +/-10 ms).

Such a sequence of periods of continuous delivery is started by a user action that requests playout to begin at a specified media time (this could be a 'play', 'seek' or 'resume' action) and continues until playout stops either due to a user action, the end of the content, or a permanent failure.

Table 30 defines the play list event metric.

Table 30: Play List

Key	Type	Description
PlayList	List	A list of playback periods. A playback period is the time interval between a user action and whichever occurs soonest of the next user action, the end of playback or a failure that stops playback.
Entry	Object	A record of a single playback period.
start	Real Time	Timestamp of the user action that starts the playback period.
mstart	Media Time	The presentation time at which playout was requested by the user action.
starttype	Enum	Type of user action which triggered playout - New playout request (e.g. initial playout or seeking) - Resume from pause - Other user request (e.g. user-requested quality change) - Start of a metrics collection period (hence earlier entries in the play list not collected)
Trace	List	List of periods of continuous rendering of decoded samples.
Traceentry	Objects	Single entry in the list.
representationid	String	The value of Representation@id from which the samples were taken. This is an optional parameter and should not be reported in case of progressive download.
subreplevel	Integer	If not present, this metric concerns the Representation as a whole. If present, subreplevel indicates the greatest value of any SubRepresentation@level being rendered. This is an optional parameter and should not be reported in case of progressive download.
start	Real Time	The time at which the first sample was rendered.
sstart	Media Time	The presentation time of the first sample rendered.
duration	Integer	The duration of the continuously presented samples (which is the same in real time and media time). 'Continuously presented' means that the media clock continued to advance at the playout speed throughout the interval.
playbackspeed	Real	The playback speed relative to normal playback speed (i.e.normal forward playback speed is 1.0).
stopreason	Enum	The reason why continuous presentation of this representation was stopped. Either: - representation switch (not relevant in case of progressive download) - rebuffering - user request - end of period - end of content - end of a metrics collection period - failure - other

NOTE: The trace may include entries for different representations that overlap in time, because multiple representations are being rendered simultaneously, for example one audio and one video representation.

10.2.8 MPD Information

This metric can be used to report Representation information from the MPD, so that reporting servers without direct access to the MPD can understand the used media characteristics.

The metric is reported whenever the client sends any other quality metrics report containing references to a Representation which MPD information has still not been reported.

Table 31 defines the MPD information for quality reporting.

Table 31: MPD Information for Quality Reporting

Key		Type	Description
MPDInformation		Object	
	representationid	String	Value of **Representation**@id for the representation addressed by the QoE metrics report.
	subreplevel	Integer	If present, value of SubRepresentation@level for the subrepresentation addressed by the QoE metrics report. If not present, the QoE metrics report concerns the representation as a whole.
	Mpdinfo	RepresentationType	Provides the MPD information for the representation or subrepresentation identified by representationid and subreplevel, if present. The following attributes and elements shall be present within mpdinfo if they are present for the identified representation or subrepresentation and their values shall be identical to those presented in the MPD: @bandwidth, @qualityRanking, @width, @height, @mimeType, and @codecs.

10.3 Quality Metrics for Progressive Download

The following metrics shall be supported by progressive download clients supporting the QoE reporting feature:

- List of HTTP Request/Response Transactions (Section 10.2.2),
- Average Throughput (Section 10.2.4),
- Initial Playout Delay (Section 10.2.5),
- Buffer Level (Section 10.2.6),
- Play List (Section 10.2.7).

10.4 Quality Metrics for DASH

The following metrics shall be supported by 3GP-DASH clients supporting the QoE reporting feature:

- List of HTTP Request/Response Transactions (Section 10.2.2),
- List of of Representation Switch Events (Section 10.2.3),
- Average Throughput (Section 10.2.4),
- Initial Playout Delay (Section 10.2.5),
- Buffer Level (Section 10.2.6),
- Play List (Section 10.2.7), and
- MPD Information (Section 10.2.8).

The @metrics attribute contains a list of quality metric keys listing all metrics that the DASH shall collect and report.

The semantics of the attributes within the **Metrics** element are provided in Table 32. The XML-syntax of a **Metrics** element is provided in Table 33.

Table 32: Semantics of Metrics element

Element or Attribute Name	Use	Description
Metrics		DASH metric element
@metrics	M	This attribute lists all quality metrics (as a list of quality metric keys as defined in section 10.2, separated by a whitespace) that the client shall report.

Element or Attribute Name	Use	Description
		Certain keys allow specifying a measurement interval or period over which a single value of the metric is derived and potentially also other parameters controlling the collection of the metrics. The parameters, if any, are included in parenthesis after the key and their semantics are specified in clause 10.2 with the metric definition itself.
Range	0..N	When specified, it indicates the time period during which quality metric collection is requested. When not present, quality metric collection is requested for the whole duration of the content.
@starttime	O	When specified, it indicates the start time of the quality metric collection operation. When not present, quality metric collection is requested from the beginning of content consumption. For services with **MPD**@type 'Live', the start time of quality metric collection can be obtained in wallclock time by adding the value of this attribute indicated in media time to the value of the **MPD**@availabilityStartTime attribute. For services with **MPD**@type 'OnDemand' , the start time is indicated in media time and is relative to the *PeriodStart* time of the first period in this MPD.
@duration	O	When specified, it indicates the duration of the quality metric collection interval. The value of this attribute is expressed in media time.
Reporting	1...N	Descriptor that provides information about the requested Quality Reporting method and formats. See clause 10.6 for the 3GP-DASH quality reporting schemes.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Elements are **bold**; attributes are non-bold and preceded with an @.		

Table 33: XML-Syntax of `Metrics` element

```
<!-- QoE Collection and Reporting -->
<xs:complexType name="MetricsType">
    <xs:sequence>
        <xs:element name="Reporting" type="DescriptorType" maxOccurs="unbounded"/>
        <xs:element name="Range" type="RangeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="metrics" type="xs:string" use="required"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<xs:complexType name="RangeType">
    <xs:sequence>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="startTime" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="duration" type="xs:duration" use="required"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

10.5 Quality Reporting Scheme for DASH

This section specifies a 3GP-DASH quality reporting scheme.

The quality reporting scheme is signaled using in the **`Reporting`** element in the **`Metrics`** element. The URN to be used for the **`Reporting`**`@schemeIdUri` shall be "`urn:3GPP:ns:PSS:DASH:QM10`".

The reporting scheme shall use the quality reporting protocol defined in section 10.6.

The semantics and XML syntax of the scheme information for the 3GP-DASH quality reporting scheme are specified in Table 34 and Table 35, respectively.

Table 34: Semantics of Quality Reporting Scheme Information

Element or Attribute Name	Use	Description
@apn	O	This attribute gives the access point that should be used for sending the QoE reports.
@format	O	This field gives the requested format for the reports. Possible formats are: 'uncompressed' and 'gzip'.
@samplepercentage	O	Percentage of the clients that should report QoE. The client uses a random number generator with the given percentage to find out if the client should report or not.
@reportingserver	M	The reporting server URL to which the reports will be sent.
@reportinginterval	O	Indicates the time(s) reports should be sent. If not present, then the client should send a report after the streaming session has ended. If present, @reportingInterval=n indicates that the client should send a report every n-th second provided that new metrics information has become available since the previous report.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>…<maxOccurs> (N=unbounded) Elements are **bold**; attributes are non-bold and preceded with an @		

Table 35: Syntax of Quality Reporting Scheme Information

```
<?xml version="1.0"?>
<xs:schema targetNamespace="urn:3GPP:ns:PSS:AdaptiveHTTPStreaming:2009:qm"
    attributeFormDefault="unqualified"
    elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns="urn:3GPP:ns:PSS:AdaptiveHTTPStreaming:2009:qm">

    <xs:annotation>
        <xs:appinfo>3GPP DASH Quality Reporting</xs:appinfo>
        <xs:documentation xml:lang="en">
            This Schema defines the quality reporting scheme information for 3GPP DASH.
        </xs:documentation>
    </xs:annotation>

    <xs:element name="ThreeGPQualityReporting" type="SimpleQualityReportingType"/>

    <xs:complexType name="SimpleQualityReportingType">
        <xs:attribute name="apn" type="xs:string" use="optional"/>
        <xs:attribute name="format" type="FormatType" use="optional"/>
        <xs:attribute name="samplePercentage" type="xs:double" use="optional"/>
        <xs:attribute name="reportingServer" type="xs:anyURI" use="required"/>
        <xs:attribute name="reportingInterval" type="xs:unsignedInt" use="optional"/>
    </xs:complexType>

    <xs:simpleType name="FormatType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="uncompressed" />
            <xs:enumeration value="gzip" />
        </xs:restriction>
    </xs:simpleType>

</xs:schema>
```

10.6 Quality Reporting Protocol

10.6.1 General

The quality reporting protocol consists of:

- The report format defined in section 10.6.2
- The reporting protocol defined in section 10.6.3

10.6.2 Report Format

The QoE report is formatted as an XML document that complies with the following XML schema:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:3gpp:metadata:2011:HSD:receptionreport"
    xmlns="urn:3gpp:metadata:2011:HSD:receptionreport" elementFormDefault="qualified">

    <xs:element name="ReceptionReport" type="ReceptionReportType"/>

    <xs:complexType name="ReceptionReportType">
        <xs:choice>
            <xs:element name="QoeReport" type="QoeReportType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="skip" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:choice>
        <xs:attribute name="contentURI" type="xs:anyURI" use="required"/>
        <xs:attribute name="clientID" type="xs:string" use="optional"/>
    </xs:complexType>

    <xs:complexType name="QoeReportType">
        <xs:sequence>
            <xs:element name="QoeMetric" type="QoeMetricType" minOccurs="1"
maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="skip" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="periodID" type="xs:string" use="required"/>
        <xs:attribute name="reportTime" type="xs:dateTime" use="required"/>
        <xs:attribute name="reportPeriod" type="xs:unsignedInt" use="required"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
    <xs:complexType name="QoeMetricType">
        <xs:choice>
            <xs:element name="HttpList" type="HttpListType"/>
            <xs:element name="RepSwitchList" type="RepSwitchListType"/>
            <xs:element name="AvgThroughput" type="AvgThroughputType" maxOccurs="unbounded"/>
            <xs:element name="InitialPlayoutDelay" type="xs:unsignedInt"/>
            <xs:element name="BufferLevel" type="BufferLevelType"/>
            <xs:element name="PlayList" type="PlayListType"/>
            <xs:element name="MPDInformation" type="MpdInformationType" maxOccurs="unbounded"/>
        </xs:choice>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:complexType name="HttpListType">
        <xs:choice>
            <xs:element name="HttpListEntry" type="HttpListEntryType" maxOccurs="unbounded"/>
        </xs:choice>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:complexType name="HttpListEntryType">
        <xs:choice>
            <xs:element name="Trace" type="HttpThroughputTraceType" maxOccurs="unbounded"/>
        </xs:choice>
        <xs:attribute name="tcpid" type="xs:unsignedInt" use="optional"/>
        <xs:attribute name="type" type="ExtensibleHttpEntryResourceType" use="optional"/>
        <xs:attribute name="url" type="xs:string" use="required"/>
        <xs:attribute name="actualUrl" type="xs:string" use="optional"/>
        <xs:attribute name="range" type="xs:string" use="optional"/>
        <xs:attribute name="trequest" type="xs:dateTime" use="required"/>
        <xs:attribute name="tresponse" type="xs:dateTime" use="required"/>
        <xs:attribute name="responsecode" type="xs:unsignedInt" use="optional"/>
        <xs:attribute name="interval" type="xs:unsignedInt" use="optional"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:simpleType name="HttpEntryResourceType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="MPD"/>
            <xs:enumeration value="MPDDeltaFile"/>
            <xs:enumeration value="XLinkExpansion"/>
```

```
            <xs:enumeration value="InitializationSegment"/>
            <xs:enumeration value="IndexSegment"/>
            <xs:enumeration value="MediaSegment"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="StringPatternType">
        <xs:restriction base="xs:string">
            <xs:pattern value="x:\S.*"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="ExtensibleHttpEntryResourceType">
        <xs:union memberTypes="HttpEntryResourceType StringPatternType"/>
    </xs:simpleType>

    <xs:complexType name="HttpThroughputTraceType">
        <xs:attribute name="s" type="xs:dateTime" use="required"/>
        <xs:attribute name="d" type="xs:unsignedInt" use="required"/>
        <xs:attribute name="b" type="UnsignedIntVectorType" use="required"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:complexType name="RepSwitchListType">
        <xs:choice>
            <xs:element name="RepSwitchEvent" type="RepSwitchEventType" maxOccurs="unbounded"/>
        </xs:choice>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:complexType name="RepSwitchEventType">
        <xs:attribute name="to" type="xs:string" use="required"/>
        <xs:attribute name="mt" type="xs:unsignedInt" use="optional"/>
        <xs:attribute name="t" type="xs:dateTime" use="optional"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:complexType name="AvgThroughputType">
        <xs:attribute name="numBytes" type="xs:unsignedInt" use="required"/>
        <xs:attribute name="activityTime" type="xs:unsignedInt" use="required"/>
        <xs:attribute name="t" type="xs:dateTime" use="required"/>
        <xs:attribute name="duration" type="xs:unsignedInt" use="required"/>
        <xs:attribute name="accessbearer" type="xs:string" use="optional"/>
        <xs:attribute name="inactivityType" type="InactivityType" use="optional"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:simpleType name="InactivityType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Pause"/>
            <xs:enumeration value="BufferControl"/>
            <xs:enumeration value="Error"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="BufferLevelType">
        <xs:choice>
            <xs:element name="BufferLevelEntry" type="BufferLevelEntryType"
maxOccurs="unbounded"/>
        </xs:choice>

        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:complexType name="BufferLevelEntryType">
        <xs:attribute name="t" type="xs:dateTime" use="required"/>
        <xs:attribute name="level" type="xs:unsignedInt" use="required"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:complexType name="PlayListType">
        <xs:choice>
            <xs:element name="Trace" type="PlayListEntryType" maxOccurs="unbounded"/>
        </xs:choice>

        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:complexType name="PlayListEntryType">
```

```
        <xs:choice>
            <xs:element name="TraceEntry" type="PlayListTraceEntryType" maxOccurs="unbounded"/>
        </xs:choice>
        <xs:attribute name="start" type="xs:dateTime" use="required"/>
        <xs:attribute name="mstart" type="xs:unsignedInt" use="required"/>
        <xs:attribute name="startType" type="StartType" use="required"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:complexType name="PlayListTraceEntryType">
        <xs:attribute name="representationId" type="xs:string" use="optional"/>
        <xs:attribute name="subrepLevel" type="xs:unsignedInt" use="optional"/>
        <xs:attribute name="start" type="xs:dateTime" use="required"/>
        <xs:attribute name="mstart" type="xs:unsignedInt" use="required"/>
        <xs:attribute name="duration" type="xs:unsignedInt" use="required"/>
        <xs:attribute name="playbackSpeed" type="xs:double" use="optional"/>
        <xs:attribute name="stopReason" type="StopReasonType" use="optional"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:simpleType name="StartType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="NewPlayoutRequst"/>
            <xs:enumeration value="Resume"/>
            <xs:enumeration value="OtherUserRequest"/>
            <xs:enumeration value="StartOfMetricsCollectionPeriod"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="StopReasonType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="RepresentationSwitch"/>
            <xs:enumeration value="Rebuffering"/>
            <xs:enumeration value="UserRequest"/>
            <xs:enumeration value="EndOfPeriod"/>
            <xs:enumeration value="EndOfContent"/>
            <xs:enumeration value="EndOfMetricsCollectionPeriod"/>
            <xs:enumeration value="Failure"/>
            <xs:enumeration value="Other"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="MpdInformationType">
        <xs:choice>
            <xs:element name="Mpdinfo" type="RepresentationType" maxOccurs="unbounded"/>
        </xs:choice>
        <xs:attribute name="representationId" type="xs:string" use="required"/>
        <xs:attribute name="subrepLevel" type="xs:unsignedInt" use="optional"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:complexType name="RepresentationType">
        <xs:attribute name="codecs" type="xs:string" use="required"/>
        <xs:attribute name="bandwidth" type="xs:unsignedInt" use="required"/>
        <xs:attribute name="qualityRanking" type="xs:unsignedInt" use="optional"/>
        <xs:attribute name="frameRate" type="xs:double" use="optional"/>
        <xs:attribute name="width" type="xs:unsignedInt" use="optional"/>
        <xs:attribute name="height" type="xs:unsignedInt" use="optional"/>
        <xs:attribute name="mimeType" type="xs:string" use="required"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:simpleType name="DoubleVectorType">
        <xs:list itemType="xs:double"/>
    </xs:simpleType>

    <xs:simpleType name="StringVectorType">
        <xs:list itemType="xs:string"/>
    </xs:simpleType>

    <xs:simpleType name="UnsignedIntVectorType">
        <xs:list itemType="xs:unsignedInt"/>
    </xs:simpleType>

</xs:schema>
```

10.6.3 Reporting Protocols

If a specific metrics server has been configured, the client shall send QoE reports using the HTTP (RFC 2616) [9] POST request carrying XML formatted metadata in its body.

An example QoE reporting based on HTTP POST request signalling is shown below:

```
POST http://www.exampleserver.com HTTP/1.1
Host: 192.68.1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0)
Content-Type: text/xml; charset=utf-8
Content-Length: 4408

<?xml version="1.0"?>
<ReceptionReport contentURI="http://www.example.com/content/content.mpd" clientID="35848574673"
xmlns="urn:3gpp:metadata:2011:HSD:receptionreport"
    xsi:schemaLocation="urn:3gpp:metadata:2011:HSD:receptionreport DASH-QoE-Report.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <QoeReport periodID="Period1" reportTime="2011-02-16T09:00:00" reportPeriod="500">
        <QoeMetric>
            <HttpList>
                <HttpListEntry type="MPD" url="http://www.example.com/content/content.mpd"
trequest="2011-02-16T08:59:30" tresponse="2011-02-16T08:59:31" interval="50">
                    <Trace s="2011-02-16T08:59:30Z" d="171" b="2367 1990 2463 1254"/>
                </HttpListEntry>
                <HttpListEntry type="InitializationSegment"
url="http://www.example.com/content/initRep1.3gp" trequest="2011-02-16T08:59:40"
tresponse="2011-02-16T08:59:41" interval="200">
                    <Trace s="2011-02-16T08:59:40.5Z" d="159" b="9345"/>
                </HttpListEntry>
                <HttpListEntry type="InitializationSegment"
url="http://www.example.com/content/initRep2.3gp" trequest="2011-02-16T08:59:41"
tresponse="2011-02-16T08:59:42" interval="200">
                    <Trace s="2011-02-16T08:59:41.5Z" d="123" b="6723"/>
                </HttpListEntry>
                <HttpListEntry type="InitializationSegment"
url="http://www.example.com/content/initRep3.3gp" trequest="2011-02-16T08:59:42"
tresponse="2011-02-16T08:59:43" interval="200">
                    <Trace s="2011-02-16T08:59:42.5Z" d="195" b="9786"/>
                </HttpListEntry>
            </HttpList>
        </QoeMetric>
        <QoeMetric>
            <InitialPlayoutDelay>10000</InitialPlayoutDelay>
        </QoeMetric>
    </QoeReport>
    <QoeReport periodID="Period1" reportTime="2011-02-16T09:08:20" reportPeriod="500">
        <QoeMetric>
            <BufferLevel>
                <BufferLevelEntry t="2011-02-16T09:08:19" level="84673"/>
                <BufferLevelEntry t="2011-02-16T09:08:20" level="93874"/>
            </BufferLevel>
        </QoeMetric>
        <QoeMetric>
            <RepSwitchList>
                <RepSwitchEvent to="Rep2"/>
                <RepSwitchEvent to="Rep3"/>
            </RepSwitchList>
        </QoeMetric>
    </QoeReport>
</ReceptionReport>
```

Annex A (informative): Example DASH Client Behaviour

A.1 Introduction

The information on client behaviour is purely informative and does not imply any normative procedures on DASH client implementations.

A.2 Overview

A 3GP-DASH client is guided by the information provided in the MPD. This example assumes that the **MPD**@type is 'dynamic'. The behaviour in case **MPD**@type being 'static' is basically a subset of the description here.

The description in this Annex assumes that the client has access to the MPD at time *FetchTime*, at its initial location if no **MPD.Location** element is present, or at a location specified in any present **MPD.Location** element. *FetchTime* is defined at the client as the time at which the server processes the request for the MPD, but should take into account delay due to MPD delivery and processing. The fetch is considered successful either if the client obtains an updated MPD or the client verifies that the MPD has not been updated since the previous fetching.

The following example client behaviour is expected to provide a continuous streaming service to the user:

1) The client parses the MPD, selects a set of Adaptation Sets suitable for its environment based on information provided in each of the **AdaptationSet** elements. The selection of Adaptation Sets may also take into account information provided by the **AdaptationSet**@group attribute.

2) Within each Adaptation Set the client selects one specific Representation, typically based on the value of the @bandwidth attribute, but also taking into account client decoding and rendering capabilities. Then it creates a list of accessible Segments for each Representation for the actual client-local time *NOW* measured in wall-clock time taking into account the procedures introduced in clause A.3.

3) The client accesses the content by requesting Segments or byte ranges of Segments. The client requests the Media Segments of the selected Representations by using the generated Segment list.

4) The client buffers media of for at least value of @minBufferTime attribute duration before starting the presentation. Then, once identified a Stream Access Point (SAP) for each of the media streams in the different Representations, it starts rendering (in wall-clock-time) of this SAP not before **MPD**@availabilityStartTime + *PeriodStart* + T_{SAP} and not after **MPD**@availabilityStartTime + *PeriodStart* + T_{SAP} + @timeShiftBufferDepth provided the observed throughput remains at or above the sum of the @bandwidth attributes of the selected Representations (if not, longer buffering may be needed). For services with **MPD**@type='dynamic', rendering the SAP at the sum of *PeriodStart* + T_{SAP} and the value of **MPD**@suggestedPresentationDelay is recommended, especially of synchronized play-out with other devices adhering to the same rule is desired.

5) Once the presentation has started, the client continues consuming the media content by continuously requesting Media Segments or parts of Media Segments . The client may switch Representations taking into account updated MPD information and/or updated information from its environment, e.g. change of observed throughout. With any request for a Media Segment containing a Stream Access Point, the client may switch to a different Representation. Seamless switching can be achieved as the different Representations are time-aligned. Advantageous switching points are announced in the MPD and/or in the Segment Index, if provided.

6) With the wall-clock time *NOW* advancing, the client consumes the available Segments. As *NOW* advances the client possibly expands the list of available Segments for each Representation according to the procedures specified in clause A.3. If the following conditions are both true, an updated MPD should be fetched:

   a) if the attribute **MPD**@minimumUpdatePeriod is present and

b) the current playback time gets within a threshold (typically described by at least the sum of the value of the `@minBufferTime` attribute and the value of the `@duration` attribute on Representation level) of the media described in the MPD for any consuming or to be consumed Representation.

7) If the clauses in 6) are true, the client should fetch a new MPD and update fetch time *FetchTime*. Once received the client now takes into account the possibly updated MPD and the new *FetchTime* in the generation of the accessible Segment Lists.

In the following a brief overview on Segment list generation, seeking, support for trick modes and switching Representations are provided.

A.3 Segment List Generation

A.3.1 General

Assume that the 3GP-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 A.1 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 synchronised with a DASH server, which is in turn is expected to be synchronised 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, 3GP-DASH clients are expected to synchronize their clocks to a globally accurate time standard.

Table A.1: Segment List

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 would restrict the URL with a byte range 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.

According to 8.4.4 there exist three different ways to describe and generate a Segment List. This description focusses 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.

Segments are available at its assigned URL if at wall-clock time *NOW* the Segment availability start time is smaller than or equal to *NOW* and the Segment availability end time is larger than or equal to *NOW*.

Furthermore, assume that for a Representation in a Period, the Segment list is indexed with *i*=1, ..., *N*.

Assume that for an MPD with fetch time *FetchTime*

- the Period start time is provided as *PeriodStart* according to clause 8.4.2 for any Period in the MPD.
- the Period end time referred as *PeriodEnd* is determined as follows: For any Period in the MPD except for the last one, the *PeriodEnd* is obtained as the value of the *PeriodStart* of the next Period. For the last Period in the MPD:
    - if the **`MPD`**`@minimumUpdatePeriod` attribute is not present, then *PeriodEnd* is defined as the end time of the Media Presentation, i.e. **`MPD`**`@availabilityStartTime` + **`MPD`**`@mediaPresentationDuration`.

if the **MPD**@minimumUpdatePeriod attribute is present, then *PeriodEnd* is defined as the smaller value of *FetchTime* + **MPD**@minimumUpdatePeriod and **MPD**@availabilityStartTime + **MPD**@mediaPresentationDuration.

The following applies for the MPD times:

- the regular duration *d* is obtained as *d* = @duration/@timescale,
- the MPD start time MediaSegment[*i*].startTime is obtained as (*i-1*)*d,
- the MPD duration MediaSegment[*i*].duration is obtained as *d* unless this Segment is the last Segment in this Period, then the MediaSegment[*i*].duration is obtained as *PeriodEnd* - MediaSegment.StartTime[*i*]).

If the @duration is not provided, then

- *N*=1,
- MediaSegment.startTime[*1*] = 0,
- MediaSegment.duration[*1*] = *PeriodEnd* - *PeriodStart*,

If the Representation contains or inherits one or more **SegmentList** elements, providing a set of explicit URL(s) for Media Segments, then all *N* Segment URLs are provided.

A.3.2 Template-based Generation of Media Segment List

If the Representation contains or inherits a **SegmentTemplate** element, then the URL of the Media Segment *i*, MediaSegment.URL[*i*], is obtained by replacing the $*Number*$ identifier by *i* + @startNumber in the **SegmentTemplate**@media string.

A.3.3 Playlist-based Generation of Media Segment List

If the Representation contains or inherits one or more **SegmentList** elements, providing a set of explicit URL(s) for Media Segments, then all *N* Segment URLs are provided.

A.3.4 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* which is greater than or equal to the *FetchTime* of the MPD.

Segments may only be accessed during their Segment availability times. Generally, Segments are only available for any time *NOW* between @availabilityStartTime and @availabilityEndTime. For times *NOW* outside this window, no Segments are available.

In addition, for services with **MPD**@type='dynamic', the Segment availability start time $T_{avail}[i]$ for a Segment *i* in a specific Period is determined as **MPD**@availabilityStartTime + *PeriodStart* + MediaSegment[*i*].startTime + MediaSegment[*i*].duration and the Segment availability end time is determined as **MPD**@availabilityStartTime + *PeriodStart* + MediaSegment[*i*].startTime + @timeshiftBufferDepth + 2*MediaSegment[*i*].duration.

In case of MPD updates, assume the variable *CheckTime* associated to an the MPD with *FetchTime* is defined as the sum of the fetch time of this operating MPD and the value of the attribute **MPD**@minimumUpdatePeriod , i.e. *CheckTime* = *FetchTime* + **MPD**@minimumUpdatePeriod. 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.

Therefore, based on an MPD that was fetched at fetch time *FetchTime* and has associated a check time *CheckTime*, the largest index i_{max} that is accessible at time *NOW* for the last Period in the MPD is $i_{max} = \max_i \{ T_{avail}[i] <= \min(\text{CheckTime}, NOW) \}$.

A.4 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 9.4.1.2, 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. 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 MPD start time MediaSegment[*i*].startTime is smaller or equal to T_M and the start of the retrieved Segment is always available.

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 *tp* and the media data for presentation time T_M 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 file, or the preceeding file 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 3GP-DASH Client needs to access Stream Access Points (SAPs). To determine the SAPs in a Media Segment in case of 3GP-DASH, the client may, for example, use the information in the Segment Index if present to locate the stream access points and the corresponding presentation time in the Media Presentation. In the case that a Segment is a 3GPP movie fragment, it is also possible for the client to use information within the "moof" and "mdat" boxes, for example, to locate SAPs 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 representation access point as the seek result. When Media Segments start with a SAP, these procedures are simplified.

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 byte range requests. By use of the Segment Index, Segment timing can be mapped to byte ranges of the Segment. By continuously using HTTP partial GET requests, only the relevant parts of the Media Segment may be accessed for improved user experience and low start-up delays.

A.5 Support for Trick Modes

The client may pause or stop a Media Presentation. In this case client simply stops requesting Media Segments or parts thereof. To resume, the client sends requests to Media Segments, starting with the next fragment after the last requested fragment.

If a specific **`Representation`** or **`SubRepresentation`** element includes the `@maxPlayoutRate` attribute, then this 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 this 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 requests.

The client may use multiple Representations to support trick mode behaviour.

A.6 Switching Representations

Based on updated 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 in the 'new' Representation at a desired presentation time T_M later than and close to the current presentation time. Presenting the 'old' Representation up to the SAP in the 'new' Representation 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 `@subsegmentStartWithSAP` the same values and conditions as above.

A.7 Reaction to Error Codes

The HTTP Streaming client provides a streaming service to the user by issuing HTTP requests for Segments at appropriate times. The HTTP Streaming client may also update the MPD by using HTTP requests. In regular operation mode, the server typically responds to such requests with status code 200 OK (for regular GET) or status code 206 Partial Content (for partial GET) and the entity corresponding to the requested resource. Other Successful 2xx or Redirection 3xx status codes may be returned.

HTTP requests may result in a Client Error 4xx or Server Error 5xx status code. Some guidelines are provided in this clause as to how an HTTP client may react to such error codes.

If the HTTP Client receives an HTTP client or server error (i.e. messages with 4xx or 5xx error code), the client should respond appropriately to the error code.

If the HTTP Client receives a repeated HTTP error for the request of an MPD, the appropriate response may involve terminating the streaming service.

If the HTTP Client receives an HTTP client error (i.e. messages with 4xx error code) for the request of an Initialization Segment, the Period containing the Initialization Segment may not be available anymore or may not be available yet. In this case the client should check if the precision of the time synchronization to a globally accurate time standard is sufficiently accurate. In case of repeated errors, the client should check for an update of the MPD.

If the HTTP Client receives an HTTP client error (i.e. messages with 4xx error code) for the request of a Media Segment, the requested Media Segment may not be available anymore or may not be available yet. In this case the client should check if the precision of the time synchronization to a globally accurate time standard is sufficiently accurate. In case of repeated errors, the client should check for an update of the MPD.

Upon receiving server errors (i.e. messages with 5xx error code), the client should check for an update of the MPD. The client may also check for alternative representations that are hosted on a different server.

A.8 Encoder Clock Drift Control

Non-alignment between the end of a Representation in one Period and the start time of the next Period may be caused by encoder clock inaccuracy. The client should align the media presentation time at each Period start. In addition, significant deviations of the start time of Segments to the media time should be detected and drift-compensating measures may be applied even before the start of the next period is reached.

Over a longer operation time, a difference in clock accuracy of the encoder and decoder may cause the playback to lag behind real-time or to interrupt temporarily due to the client trying to access data faster than real-time. Clients may avoid these anomalies by using the Producer Reference Time boxes as defined in clause G.5 as follows. The pace r1 of

the encoder clock in relation to the UTC is recovered from Producer Reference Time boxes. If the relative pace r1 is less than 1, equal to 1, or greater than 1, the encoder clock runs more slowly than the UTC, at an identical pace compared to the UTC, or faster than the UTC, respectively. The pace r2 of the receiver playout clock in relation to UTC is created by accessing a UTC source. A timescale multiplication factor c is equal to r1/r2. A presentation time on a timeline of the receiver playout clock is derived for each sample or access unit by multiplying the composition time of the sample (as indicated by the file format structures) or the presentation time of the access unit (as indicated by the respective Program Elementary Stream header) by the timescale multiplication factor c.

Annex B (normative): Media Presentation Description Schema

B.1 Introduction

The main schema is provided in Annex B.2 in Table B-1. The main schema refers to the extension schema in Annex B.3 and in section 8.3.

B.2 Main Schema

Table B-1: XML schema of the MPD

```
<?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:x3gpp="urn:3GPP:ns:DASH:MPD-ext:2011"
  xmlns="urn:mpeg:dash:schema:mpd:2011">

  <xs:import namespace="http://www.w3.org/1999/xlink" schemaLocation=" xlink.xsd"/>
  <xs:import namespace="urn:3GPP:ns:DASH:MPD-ext:2011" schemaLocation="3gpp-2011.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="x3gpp:DeltaSupport" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string"/>
    <xs:attribute name="profiles" type="xs:string" use="required"/>
    <xs:attribute name="type" type="PresentationType" default="static"/>
    <xs:attribute name="availabilityStartTime" type="xs:dateTime"/>
    <xs:attribute name="availabilityEndTime" type="xs:dateTime"/>
    <xs:attribute name="mediaPresentationDuration" type="xs:duration"/>
    <xs:attribute name="minimumUpdatePeriod" type="xs:duration"/>
    <xs:attribute name="minBufferTime" type="xs:duration" use="required"/>
    <xs:attribute name="timeShiftBufferDepth" type="xs:duration"/>
    <xs:attribute name="suggestedPresentationDelay" type="xs:duration"/>
    <xs:attribute name="maxSegmentDuration" type="xs:duration"/>
    <xs:attribute name="maxSubsegmentDuration" type="xs:duration"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <!-- Presentation Type enumeration -->
  <xs:simpleType name="PresentationType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="static"/>
      <xs:enumeration value="dynamic"/>
    </xs:restriction>
```

```
  </xs:simpleType>

  <!-- Period -->
  <xs:complexType name="PeriodType">
    <xs:sequence>
      <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="SegmentBase" type="SegmentBaseType" minOccurs="0"/>
      <xs:element name="SegmentList" type="SegmentListType" minOccurs="0"/>
      <xs:element name="SegmentTemplate" type="SegmentTemplateType" minOccurs="0"/>
      <xs:element name="AdaptationSet" type="AdaptationSetType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute ref="xlink:href"/>
    <xs:attribute ref="xlink:actuate" default="onRequest"/>
    <xs:attribute name="id" type="xs:string" />
    <xs:attribute name="start" type="xs:duration"/>
    <xs:attribute name="duration" type="xs:duration"/>
    <xs:attribute name="bitstreamSwitching" type="xs:boolean" default="false"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <!-- Adaptation Set -->
  <xs:complexType name="AdaptationSetType">
    <xs:complexContent>
      <xs:extension base="RepresentationBaseType">
        <xs:sequence>
          <xs:element name="Accessibility" type="DescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
          <xs:element name="Role" type="DescriptorType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="Rating" type="DescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
          <xs:element name="Viewpoint" type="DescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
          <xs:element name="ContentComponent" type="ContentComponentType" minOccurs="0"
maxOccurs="unbounded"/>
          <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="SegmentBase" type="SegmentBaseType" minOccurs="0"/>
          <xs:element name="SegmentList" type="SegmentListType" minOccurs="0"/>
          <xs:element name="SegmentTemplate" type="SegmentTemplateType" minOccurs="0"/>
          <xs:element name="Representation" type="RepresentationType" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute ref="xlink:href"/>
        <xs:attribute ref="xlink:actuate" default="onRequest"/>
        <xs:attribute name="id" type="xs:unsignedInt"/>
        <xs:attribute name="group" type="xs:unsignedInt"/>
        <xs:attribute name="lang" type="xs:language"/>
        <xs:attribute name="contentType" type="xs:string"/>
        <xs:attribute name="minBandwidth" type="xs:unsignedInt"/>
        <xs:attribute name="maxBandwidth" type="xs:unsignedInt"/>
        <xs:attribute name="minWidth" type="xs:unsignedInt"/>
        <xs:attribute name="maxWidth" type="xs:unsignedInt"/>
        <xs:attribute name="minHeight" type="xs:unsignedInt"/>
        <xs:attribute name="maxHeight" type="xs:unsignedInt"/>
        <xs:attribute name="minFrameRate" type="FrameRateType"/>
        <xs:attribute name="maxFrameRate" type="FrameRateType"/>
        <xs:attribute name="segmentAlignment" type="ConditionalUintType" default="false"/>
        <xs:attribute name="subsegmentAlignment" type="ConditionalUintType" default="false"/>
        <xs:attribute name="subsegmentStartsWithSAP" type="SAPType" default="0"/>
        <xs:attribute name="bitstreamSwitching" type="xs:boolean"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Type for Frame Rate -->
  <xs:simpleType name="FrameRateType">
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9]*[0-9](/[0-9]*[0-9])?"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- Conditional Unsigned Integer (unsignedInt or boolean) -->
  <xs:simpleType name="ConditionalUintType">
    <xs:union memberTypes="xs:unsignedInt xs:boolean"/>
  </xs:simpleType>
```

```
  <!-- Content Component -->
  <xs:complexType name="ContentComponentType">
    <xs:sequence>
      <xs:element name="Accessibility" type="DescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="Role" type="DescriptorType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Rating" type="DescriptorType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Viewpoint" type="DescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:unsignedInt"/>
    <xs:attribute name="lang" type="xs:language"/>
    <xs:attribute name="contentType" type="xs:string"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <!-- Representation -->
  <xs:complexType name="RepresentationType">
    <xs:complexContent>
      <xs:extension base="RepresentationBaseType">
        <xs:sequence>
          <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="SubRepresentation" type="SubRepresentationType" minOccurs="0"
maxOccurs="unbounded"/>
          <xs:element name="SegmentBase" type="SegmentBaseType" minOccurs="0"/>
          <xs:element name="SegmentList" type="SegmentListType" minOccurs="0"/>
          <xs:element name="SegmentTemplate" type="SegmentTemplateType" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="id" type="StringNoWhitespaceType" use="required"/>
        <xs:attribute name="bandwidth" type="xs:unsignedInt" use="required"/>
        <xs:attribute name="qualityRanking" type="xs:unsignedInt"/>
        <xs:attribute name="mediaStreamStructureId" type="StringVectorType"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- String without white spaces -->
  <xs:simpleType name="StringNoWhitespaceType">
    <xs:restriction base="xs:string">
      <xs:pattern value="[^\r\n\t \p{Z}]*"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- SubRepresentation -->
  <xs:complexType name="SubRepresentationType">
    <xs:complexContent>
      <xs:extension base="RepresentationBaseType">
        <xs:attribute name="level" type="xs:unsignedInt"/>
        <xs:attribute name="dependencyLevel" type="UIntVectorType"/>
        <xs:attribute name="bandwidth" type="xs:unsignedInt"/>
        <xs:attribute name="contentComponent" type="StringVectorType"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Representation base (common attributes and elements) -->
  <xs:complexType name="RepresentationBaseType">
    <xs:sequence>
      <xs:element name="AudioChannelConfiguration" type="DescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="ContentProtection" type="DescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="EssentialProperty" type="DescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="SupplementalProperty" type="DescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="profiles" type="xs:string"/>
    <xs:attribute name="width" type="xs:unsignedInt"/>
    <xs:attribute name="height" type="xs:unsignedInt"/>
    <xs:attribute name="frameRate" type="FrameRateType"/>
    <xs:attribute name="audioSamplingRate" type="xs:string"/>
    <xs:attribute name="mimeType" type="xs:string"/>
    <xs:attribute name="codecs" type="xs:string"/>
    <xs:attribute name="maximumSAPPeriod" type="xs:double"/>
    <xs:attribute name="startWithSAP" type="SAPType"/>
    <xs:attribute name="maxPlayoutRate" type="xs:double"/>
```

```
    <xs:attribute name="codingDependency" type="xs:boolean"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <!-- Stream Access Point type enumeration -->
  <xs:simpleType name="SAPType">
    <xs:restriction base="xs:unsignedInt">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="6"/>
    </xs:restriction>
  </xs:simpleType>


  <!-- Segment information base -->
  <xs:complexType name="SegmentBaseType">
    <xs:sequence>
      <xs:element name="Initialization" type="URLType" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="timescale" type="xs:unsignedInt"/>
    <xs:attribute name="presentationTimeOffset" type="xs:unsignedLong"/>
    <xs:attribute name="timeShiftBufferDepth" type="xs:duration"/>
    <xs:attribute name="indexRange" type="xs:string"/>
    <xs:attribute name="indexRangeExact" type="xs:boolean" default="false"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <!-- Multiple Segment information base -->
  <xs:complexType name="MultipleSegmentBaseType">
    <xs:complexContent>
      <xs:extension base="SegmentBaseType">
        <xs:attribute name="duration" type="xs:unsignedInt"/>
        <xs:attribute name="startNumber" type="xs:unsignedInt"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Segment Info item URL/range -->
  <xs:complexType name="URLType">
    <xs:sequence>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="sourceURL" type="xs:anyURI"/>
    <xs:attribute name="range" type="xs:string"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <!-- Segment List -->
  <xs:complexType name="SegmentListType">
    <xs:complexContent>
      <xs:extension base="MultipleSegmentBaseType">
        <xs:sequence>
          <xs:element name="SegmentURL" type="SegmentURLType" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute ref="xlink:href"/>
        <xs:attribute ref="xlink:actuate" default="onRequest"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Segment URL  -->
  <xs:complexType name="SegmentURLType">
    <xs:sequence>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="media" type="xs:anyURI"/>
    <xs:attribute name="mediaRange" type="xs:string"/>
    <xs:attribute name="indexRange" type="xs:string"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <!-- Segment Template -->
  <xs:complexType name="SegmentTemplateType">
    <xs:complexContent>
      <xs:extension base="MultipleSegmentBaseType">
        <xs:attribute name="media" type="xs:string"/>
        <xs:attribute name="initialization" type="xs:string"/>
```

```
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Whitespace-separated list of strings -->
  <xs:simpleType name="StringVectorType">
    <xs:list itemType="xs:string"/>
  </xs:simpleType>

  <!-- Whitespace-separated list of unsigned integers -->
  <xs:simpleType name="UIntVectorType">
    <xs:list itemType="xs:unsignedInt"/>
  </xs:simpleType>

  <!-- Base URL -->
  <xs:complexType name="BaseURLType">
    <xs:simpleContent>
      <xs:extension base="xs:anyURI">
        <xs:attribute name="serviceLocation" type="xs:string"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <!-- Program Information -->
  <xs:complexType name="ProgramInformationType">
    <xs:sequence>
      <xs:element name="Title" type="xs:string" minOccurs="0"/>
      <xs:element name="Source" type="xs:string" minOccurs="0"/>
      <xs:element name="Copyright" type="xs:string" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="lang" type="xs:language"/>
    <xs:attribute name="moreInformationURL" type="xs:anyURI"/>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <!-- Descriptor -->
  <xs:complexType name="DescriptorType">
    <xs:sequence>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="schemeIdUri" type="xs:anyURI" use="required"/>
    <xs:attribute name="value" type="xs:string"/>
    <xs:attribute name="id" type="xs:string"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <!-- Metrics -->
  <xs:complexType name="MetricsType">
    <xs:sequence>
      <xs:element name="Reporting" type="DescriptorType" maxOccurs="unbounded"/>
      <xs:element name="Range" type="RangeType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="metrics" type="xs:string" use="required"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <!-- Metrics Range -->
  <xs:complexType name="RangeType">
    <xs:attribute name="starttime" type="xs:duration"/>
    <xs:attribute name="duration" type="xs:duration"/>
  </xs:complexType>

</xs:schema>
```

B.3 3GPP Extension Schema

Table B-2: XML schema of the 3GPP Extensions for MPD

```
<?xml version="1.0"?>
<xs:schema targetNamespace="urn:3GPP:ns:DASH:MPD-ext:2011"
    attributeFormDefault="unqualified" elementFormDefault="qualified"
```

```
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="urn:3GPP:ns:DASH:MPD-ext:2011">
    <xs:annotation>
        <xs:appinfo>Extensions to Media Presentation Description for 3GPP</xs:appinfo>
    </xs:annotation>

    <xs:element name="DeltaSupport" type="DeltaSupportType"/>

    <!--DeltaSupport for the MPD -->
    <xs:complexType name="DeltaSupportType">
        <xs:sequence>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="sourceURL" type="xs:anyURI" use="required"/>
        <xs:attribute name="availabilityDuration" type="xs:duration"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>


</xs:schema>
```

Annex C (normative): Descriptor Scheme Definitions

C.1 Introduction

This annex defines descriptors that are defined in this specification. In particular the following descriptors are defined

- Role descriptor scheme in clause C.2.

C.2 Role Descriptor Scheme

The URN "urn:mpeg:dash:role:2011" is defined to identify the role scheme defined in Table C.1. Note that **Role**@value shall be assigned to Adaptation Sets that contain a media component type to which this role is associated.

Table C.1 — Role@value attríbute for scheme with a value "urn:mpeg:dash:role:2011"

Role@value	**Description**
caption	captions (see note 3 below)
subtitle	subtitles (see note 3 below)
main	main media component(s) which is/are intended for presentation if no other information is provided
alternate	media content component(s) that is/are an alternative to (a) main media content component(s) of the same media component type (see note 2 below)
supplementary	media content component that is supplementary to a media content component of a different media component type (see Note 1 below)
commentary	media content component with commentary (e.g. director"s commentary) (typically audio)
dub	media content component which is presented in a different language from the original (e.g. dubbed audio, translated captions)

NOTES

1) A normal audio/video program labels both the primary audio and video as "main". However, when the two media component types are not equally important, for example (a) video providing a pleasant visual experience to accompany a music track that is the primary content or (b) ambient audio accompanying a video showing a live scene such as a sports event, that is the primary content, the accompanying media may be assigned a "supplementary" role.

2) alternate media content components should carry other descriptors to indicate in what way it differs from the main media content components (e.g. a Viewpoint descriptor or a Role descriptor), especially when multiple alternate media content components including multiple supplementary media content components are available.

3) open ('burned in') captions or subtitles would be marked as media type component "video" only, but having a descriptor saying 'caption' or 'subtitle';

Annex D (informative): MPD Examples

D.1 On-Demand Service

Table D.1 provides an example MPD for an On-Demand service.

Table D.1: Example MPD for an On-Demand Service

```
<?xml version="1.0"?>
<MPD
    profiles="urn:3GPP:PSS:profile:DASH10"
    type="static"
    minBufferTime="PT10S"
    mediaPresentationDuration="PT2H"
    availabilityStartTime="2010-04-01T09:30:47Z"
    availabilityEndTime="2010-04-07T09:30:47Z"
    xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 3GPP-Rel10-MPD.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="urn:mpeg:dash:schema:mpd:2011">
    <ProgramInformation moreInformationURL="http://www.example.com">
        <Title>Example</Title>
    </ProgramInformation>
    <BaseURL>http://www.example.com</BaseURL>
    <Period start="PT0S">
        <AdaptationSet mimeType="video/3gpp">
            <ContentComponent contentType="video"/>
            <ContentComponent contentType="audio" lang="en"/>
            <Representation codecs="s263, samr" bandwidth="256000" id="256">
                <BaseURL>"rep1"</BaseURL>
                <SegmentList duration="1000" timescale="100">
                    <Initialization sourceURL="seg-init.3gp"/>
                    <SegmentURL media="seg-1.3gp"/>
                    <SegmentURL media="seg-2.3gp"/>
                    <SegmentURL media="seg-3.3gp"/>
                </SegmentList>
            </Representation>
            <Representation codecs="mp4v.20.9, mp4a.E1" bandwidth="128000" id="128">
                <BaseURL>"rep2"</BaseURL>
                <SegmentList duration="10">
                    <Initialization sourceURL="seg-init.3gp"/>
                    <SegmentURL media="seg-1.3gp"/>
                    <SegmentURL media="seg-2.3gp"/>
                    <SegmentURL media="seg-3.3gp"/>
                </SegmentList>
            </Representation>
        </AdaptationSet>
    </Period>
    <Period start="PT30S">
        <SegmentTemplate
            duration="10"
            initialization="seg-init-$RepresentationId$.3gp"
            media="http://example.com/$RepresentationId$/$Number$.3gp"/>
        <AdaptationSet mimeType="video/3gpp" codecs="mp4v.20.9, mp4a.E1">
            <ContentComponent contentType="video"/>
            <ContentComponent contentType="audio" lang="en"/>
            <Representation bandwidth="256000" id="1"/>
            <Representation bandwidth="128000" id="2"/>
        </AdaptationSet>
    </Period>
</MPD>
```

D.2 Live Service

Table D.2 provides an example MPD for a live service.

Table D.2: Example MPD for a Live Service

```
<?xml version="1.0"?>
<MPD
    profiles="urn:3GPP:PSS:profile:DASH10"
    type="dynamic"
    minBufferTime="PT3S"
    availabilityStartTime="2010-04-26T08:45:00-08:00"
    minimumUpdatePeriod="PT5M0S"
    timeShiftBufferDepth="PT1H30M0S"
    xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 3GPP-Rel10-MPD.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="urn:mpeg:dash:schema:mpd:2011">
    <ProgramInformation moreInformationURL="http://www.example.com">
        <Title>Example 3: 3GPP SA4 Meeting in Vancouver as Live Broadcast</Title>
        <Source>3GPP</Source>
    </ProgramInformation>
    <Period start="PT0S" id="0">
        <AdaptationSet mimeType='video/3gpp' codecs="avc1.42E00B" width="320" height="240"
contentType="video">
            <SegmentTemplate
                duration="60"
                initialization="http://www.ad-server.com/1-day-black/QVGA/0.3gp"
                media="http://www.ad-server.com/1-day-black/QVGA/$Number$.3gp">
            </SegmentTemplate>
            <Representation id="Ad-QVGA" bandwidth="10000">
        </Representation>
        </AdaptationSet>
    </Period>
    <Period start="PT15M0S" id="1">
        <SegmentTemplate
            duration="10"
            initialization="http://www.example.com/Period-2010-04-26T08-45-00/rep-
$RepresentationID$/seg-0.3gp"
            media="http://www.example.com/Period-2010-04-26T08-45-00/rep-
$RepresentationID$/seg-$Number$.3gp"/>
        <AdaptationSet mimeType='video/3gpp'>
            <ContentComponent contentType="video"/>
            <ContentComponent contentType="audio" lang="en"/>
            <Representation
id="QVGA-LQ" codecs="avc1.42E00C, mp4a.40.2" bandwidth="192000" width="320" height="240"/>
            <Representation id="QVGA-HQ" codecs="avc1.42E00C, mp4a.40.2" bandwidth="384000"
width="320" height="240"/>        <Representation id="VGA-LQ" mimeType='video/3gpp'
codecs="avc1.64001E, mp4a.40.2" bandwidth="512000" width="640" height="480"/>
            <Representation id="VGA-HQ" codecs="avc1.64001E, mp4a.40.2" bandwidth="1024000"
width="640" height="480"/>
        </AdaptationSet>
    </Period>
    <Period start="PT2H01M22.12S" id="2">
        <SegmentTemplate duration="10"
            media="http://www.ad-server.com/15min-Ads/$RepresentationID$/$Number$.3gp"
            initialization="http://www.ad-server.com/15min-Ads/$RepresentationID$/0.3gp"/>
        <AdaptationSet mimeType='video/3gpp'>
            <ContentComponent contentType="video"/>
            <ContentComponent contentType="audio" lang="en"/>
            <Representation id="QVGA" codecs="avc1.42E00C, mp4a.40.2" bandwidth="256000"
width="320" height="240"/>
            <Representation id="VGA"  codecs="avc1.64001E, mp4a.40.2" bandwidth="512000"
width="640" height="480"/>
        </AdaptationSet>
    </Period>
    <Period start="PT2H16M22.12S" id="3">
        <SegmentTemplate
            duration="10"
            media="http://www.example.com/Period-2010-04-26T11-01-22/rep-
$RepresentationID$/seg-$Number$.3gp"
            initialization="http://www.example.com/Period-2010-04-26T11-01-22/rep-
$RepresentationID$/seg-0.3gp"
        />
        <AdaptationSet mimeType='video/3gpp' contentType="video">
            <Representation id="QVGA-LQ" codecs="avc1.42E00C" bandwidth="192000" width="320"
height="240"/>
            <Representation id="QVGA-HQ" codecs="avc1.42E00C" bandwidth="384000" width="320"
height="240"/>
            <Representation id="VGA-LQ" codecs="avc1.64001E" bandwidth="512000" width="640"
height="480"/>
            <Representation id="VGA-HQ" codecs="avc1.64001E" bandwidth="1024000" width="640"
height="480"/>
```

```
        </AdaptationSet>
        <AdaptationSet mimeType='audio/3gpp' contentType="audio" lang="en">
            <Representation id="audio" codecs="mp4a.40.2" bandwidth="32000"/>
            <Representation id="audio" codecs="mp4a.40.2" bandwidth="64000"/>
        </AdaptationSet>
    </Period>
</MPD>
```

D.3 MPD Assembly

Table D.3 provides an example MPD with reference to external Period element as provided in Table D.4. An equivalent MPD to the one in Table D.3 after dereferencing with the **`Period`** element in Table D.4 is shown in Table D.2.

Table D.3: Example MPD with reference to external Period element

```
<?xml version="1.0"?>
<MPD
    profiles="urn:3GPP:PSS:profile:DASH10"
    type="dynamic"
    minBufferTime="PT3S"
    availabilityStartTime="2010-04-26T08:45:00-08:00"
    minimumUpdatePeriod="PT5M0S"
    timeShiftBufferDepth="PT1H30M0S"
    xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 3GPP-Rel10-MPD.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns="urn:mpeg:dash:schema:mpd:2011">
    <ProgramInformation moreInformationURL="http://www.example.com">
        <Title>Example 3: 3GPP SA4 Meeting in Vancouver as Live Broadcast</Title>
        <Source>3GPP</Source>
    </ProgramInformation>
    <Period start="PT0S" id="0">
        <AdaptationSet mimeType='video/3gpp' codecs="avc1.42E00B" width="320" height="240"
contentType="video">
            <SegmentTemplate
                duration="60"
                initialization="http://www.ad-server.com/1-day-black/QVGA/0.3gp"
                media="http://www.ad-server.com/1-day-black/QVGA/$Number$.3gp">
            </SegmentTemplate>
            <Representation id="Ad-QVGA" bandwidth="10000">
            </Representation>
        </AdaptationSet>
    </Period>
    <Period start="PT15M0S" id="1">
        <SegmentTemplate
            duration="10"
            initialization="http://www.example.com/Period-2010-04-26T08-45-00/rep-
$RepresentationID$/seg-0.3gp"
            media="http://www.example.com/Period-2010-04-26T08-45-00/rep-
$RepresentationID$/seg-$Number$.3gp"/>
        <AdaptationSet mimeType='video/3gpp'>
            <ContentComponent contentType="video"/>
            <ContentComponent contentType="audio" lang="en"/>
            <Representation id="QVGA-LQ" codecs="avc1.42E00C, mp4a.40.2" bandwidth="192000"
width="320" height="240"/>
            <Representation id="QVGA-HQ" codecs="avc1.42E00C, mp4a.40.2" bandwidth="384000"
width="320" height="240"/>        <Representation id="VGA-LQ" mimeType='video/3gpp'
codecs="avc1.64001E, mp4a.40.2" bandwidth="512000" width="640" height="480"/>
            <Representation id="VGA-HQ" codecs="avc1.64001E, mp4a.40.2" bandwidth="1024000"
width="640" height="480"/>
        </AdaptationSet>
    </Period>
    <Period xlink:ref="http://www.example.com/Period.xml" id="2"/>
    <Period start="PT2H16M22.12S" id="3">
        <SegmentTemplate
            duration="10"
            media="http://www.example.com/Period-2010-04-26T11-01-22/rep-
$RepresentationID$/seg-$Number$.3gp"
            initialization="http://www.example.com/Period-2010-04-26T11-01-22/rep-
$RepresentationID$/seg-0.3gp"
        />
        <AdaptationSet mimeType='video/3gpp' contentType="video">
            <Representation id="QVGA-LQ" codecs="avc1.42E00C" bandwidth="192000" width="320"
height="240"/>
```

```
            <Representation id="QVGA-HQ" codecs="avc1.42E00C" bandwidth="384000" width="320"
height="240"/>
            <Representation id="VGA-LQ" codecs="avc1.64001E" bandwidth="512000" width="640"
height="480"/>
            <Representation id="VGA-HQ" codecs="avc1.64001E" bandwidth="1024000" width="640"
height="480"/>
        </AdaptationSet>
        <AdaptationSet mimeType='audio/3gpp' contentType="audio" lang="en">
            <Representation id="audio" codecs="mp4a.40.2" bandwidth="32000"/>
            <Representation id="audio" codecs="mp4a.40.2" bandwidth="64000"/>
        </AdaptationSet>
    </Period>
</MPD>
```

Table D.4: External Period

```
<?xml version="1.0"?>
<Period start="PT15M0S">
    <SegmentTemplate
        duration="10"
        initialization="http://www.example.com/Period-2010-04-26T08-45-00/rep-
$RepresentationID$/seg-0.3gp"
        media="http://www.example.com/Period-2010-04-26T08-45-00/rep-$RepresentationID$/seg-
$Number$.3gp"/>
    <AdaptationSet mimeType='video/3gpp'>
        <ContentComponent contentType="video"/>
        <ContentComponent contentType="audio" lang="en"/>
        <Representation id="QVGA-LQ" codecs="avc1.42E00C, mp4a.40.2" bandwidth="192000"
width="320" height="240"/>
        <Representation id="QVGA-HQ" codecs="avc1.42E00C, mp4a.40.2" bandwidth="384000"
width="320" height="240"/>
        <Representation id="VGA-LQ" mimeType='video/3gpp' codecs="avc1.64001E, mp4a.40.2"
bandwidth="512000" width="640" height="480"/>
        <Representation id="VGA-HQ" codecs="avc1.64001E, mp4a.40.2" bandwidth="1024000"
width="640" height="480"/>
    </AdaptationSet>
</Period>
```

D.4 MPD Deltas

In the following MPD example, the content is 30 minutes in duration. There are 3 Periods, each of 10 minutes duration. Each Period has 3 Representations and each Representation is contained within one 3gp file. Each Representation has audio encoded with Low Complexity-AAC. One Representation of each Period (p1rep1.3gp, p2rep1.3gp, and p3rep1.3gp) has video resolution 320x240 encoded with H.264 baseline profile level 1.1. Another Representation of each Period (p1rep2.3gp, p2rep2.3gp, and p3rep2.3gp) has resolution 320x240 encoded with H.264 baseline profile level 1.3. Finally, a third representation in each period (p1rep3.3gp, p2rep3.3gp, and p3rep3.3gp) has resolution 480x240 encoded with H.264 baseline profile level 2.1. One Representation of each Period has bandwidth of 239 kbps, a second representation has bandwidth of 478 kbps, and a third representation has bandwidth of 892 kbps.

Since each represention is contained in one file, the Initialization Segments and the Media Segments for a representation are accessed with byte ranges. Each **`SegmentUrl`** element in the MPD contains a **`mediarange`** attribute and the corresponding byte range for the Initialization Segment or Media Segment. For the example each Segment of all representations is 10 seconds in duration.

Line numbers of the MPD in the example are shown for clarity, although these would not be present in the MPD.

EXAMPLE 1 (add)

The change of adding the "Url" element for the last Segment to the Representation of the third Period with 239K bandwidth can be described as

```
517a
                <SegmentUrl mediarange="17339554-17642841"/>
.
```

The line number of the MPD where the delta is applied is 517. The following line is added:

EXAMPLE 2 (replace)

Consider the change of replacing the line containing the **`DeltaSupport`** element in the next MPD.

```
20c
      <DeltaSupport sourceURL="delta2.mpdd" availabilityDuration="120s"/>
.
```

EXAMPLE 3(delete)

If lines 8 through 10 of the original MPD are deleted and not present in the updated MPD, the delta to express this is:

```
8,10d
.
```

Below is what the MPD looks like after 30 minutes. In this case, the MPD is updated approximately every 10 seconds.

```
1<?xml version="1.0"?>
2<MPD
        profiles="urn:3GPP:PSS:profile:DASH10"
3       type="dynamic"
4       availabilityStartTime="2010-07-01T05:00:00Z"
5       availabilityEndTime="2010-07-08T05:00:00Z"
6       mediaPresentationDuration="PT2H"
7       minimumUpdatePeriod="PT10S"
8       minBufferTime="PT10S"
9       timeShiftBufferDepth="PT30M"
10      baseUrl="http://www.example.com/"
11      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
12      xmlns:x3gpp="urn:3GPP:ns:DASH:MPD-ext:2011"
13      xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 3GPP-2011.xsd"
14      xmlns="urn:mpeg:dash:schema:mpd:2011">
15
16 <ProgramInformation moreInformationURL="http://www.example.com">
17  <Title>Example</Title>
18  <Source>Example</Source>
19  <Copyright>Example</Copyright>
20 </ProgramInformation>
21 <x3gpp:DeltaSupport sourceURL="delta1.mpdd" availabilityDuration="120s"/>
22 <Period start="PT0S" bitstreamSwitching="true" id="0">
23  <AdaptationSet mimeType="video/3gpp">
24    <ContentComponent contentType="video"/>
25    <ContentComponent contentType="audio" lang="en"/>
26    <Representation id="0" bandwidth="239000" width="320" height="240" codecs="avc1.42E00b,
mp4a.40.2">
         <BaseURL>"p1rep1.3gp"</BaseURL>
27       <SegmentList duration="10" >
28          <Initialization range="0-985" />
29                              <SegmentUrl  mediarange="986-293761" />
30                              <SegmentUrl  mediarange="293762-592501" />
                                                    .
                                                    .
                                                    .

84                              <SegmentUrl  mediarange="17600065-17894640" />
85       </SegmentList>
86    </Representation>
87    <Representation id="1" bandwidth="478000" width="320" height="240" codecs="avc1.42E00d,
mp4a.40.2" >
88       <BaseURL>"p1rep2.3gp"</BaseURL>
89       <SegmentList duration="10">
90          <Initialization range="0-985" />
91                              <SegmentUrl  mediarange="986-586538" />
92                              <SegmentUrl  mediarange="586539-1184019" />
                                                    .
                                                    .
                                                    .
149                             <SegmentUrl  mediarange="35199171-35788323" />
150      </SegmentList>
151    </Representation>
152    <Representation id="2" bandwidth="892000" width="480" height="240"
codecs="avc1.42E015, mp4a.40.2">
153      <BaseURL>"p1rep3.3gp"</BaseURL>
154      <SegmentList duration="10">
155            <Initialization range="0-985" />
```

```
156                             <SegmentUrl  mediarange="986-1093691" />
157                             <SegmentUrl  mediarange="1093692-2208656" />
                                                    .
                                                    .
                                                    .
214                             <SegmentUrl  mediarange="65684646-66784068" />
215      </SegmentList>
216   </Representation>
216 </AdaptationSet>
217</Period>
218<Period start="PT10M0S" bitstreamSwitching="true" id="1">
    <AdaptationSet mimeType="video/3gpp">
      <ContentComponent contentType="video"/>
      <ContentComponent contentType="audio" lang="en"/>
219 <Representation  id="0" bandwidth="239000" width="320" height="240" codecs="avc1.42E00b,
mp4a.40.2">
         <BaseURL>"p2rep0.3gp"</BaseURL>
220      <SegmentList duration="10">
221         <Initialization range="0-985" />
222                             <SegmentUrl  mediarange="986-296011" />
223                             <SegmentUrl  mediarange="296012-595787" />
                                                    .
                                                    .
                                                    .

280                              <SegmentUrl  mediarange="17647666-17946154" />
281      </SegmentList>
282 </Representation>
283 <Representation id="1" bandwidth="478000" width="320" height="240" codecs="avc1.42E00d,
mp4a.40.2">

284      <SegmentList duration="10">
285         <BaseURL>"p2rep1.3gp"</BaseURL>
286         <Initialization range="0-985" />
287                             <SegmentUrl  mediarange="986-591037" />
288                             <SegmentUrl  mediarange="591038-1190590" />
                                                    .
                                                    .
                                                    .
385                             <SegmentUrl  mediarange="35294377-35891354" />
386      </SegmentList>
387  </Representation>
388  <Representation id="2" bandwidth="892000" width="480" height="240" codecs="avc1.42E015,
mp4a.40.2">
389      <BaseURL>"p2rep2.3gp"</BaseURL>
390      <SegmentList duration="PT10S">
391         <Initialization range="0-985" />
392                             <SegmentUrl mediarange="986-1102088" />
393                             <SegmentUrl mediarange="1102089-2220920" />
                                                    .
                                                    .
                                                    .
450                             <SegmentUrl mediarange="65862331-66976355" />
451      </SegmentList>
452   </Representation>
453  </AdaptationSet>
454</Period>
455<Period start="PT20M0S" bitstreamSwitching="true">
456 <AdaptationSet mimeType="video/3gpp">
457   <ContentComponent contentType="video"/>
458   <ContentComponent contentType="audio" lang="en"/>
459   <Representation  id="0" bandwidth="239000" width="320" height="240"
codecs="avc1.42E00b, mp4a.40.2">
460       <BaseURL>"p3rep0.3gp"</BaseURL>
461      <SegmentList duration="10">
462         <Initialization range="0-985" />
463                             <SegmentUrl  mediarange="986-302469" />
464                             <SegmentUrl  mediarange="302470-597839" />
                                                    .
                                                    .
                                                    .
517                             <SegmentUrl  mediarange="17040002-17339553" />
518      </SegmentList>
519   </Representation>
520   <Representation id="1" bandwidth="478000" width="320" height="240" codecs="avc1.42E00d,
mp4a.40.2" >
521      <BaseURL>"p3rep1.3gp"</BaseURL>
522      <SegmentList duration="10">
```

```
523         <Initialization range="0-985" />
524                             <SegmentUrl  mediarange="986-603953" />
525                             <SegmentUrl  mediarange="603954-1194693" />
                                                    .
                                                    .
                                                    .
582                             <SegmentUrl  mediarange="34079046-34678149" />
583      </SegmentList>
584   </Representation>
585   <Representation id="2" bandwidth="892000" width="480" height="240" codecs="avc1.42E015,
mp4a.40.2" >
586      <BaseURL>"p3rep2.3gp"</BaseURL>
587      <SegmentList duration="10">
588         <Initialization range="0-985" />
589                             <SegmentUrl  mediarange="986-1126190" />
590                             <SegmentUrl  mediarange="1126191-2228575" />
                                                    .
                                                    .
                                                    .
647                              <SegmentUrl  mediarange="63594383-64712374" />
648      </SegmentList>
649   </Representation>
650  </AdaptationSet>
651 </Period>
652</MPD>
```

Since the value of `@sourceURL` in the above MPD is '`delta1.mpdd`', `delta1.mpdd` is an empty file at the time of publication of the above MPD.

The following file is `delta1.mpdd` after the next MPD update. Notice that clients have access to the new value of `@sourceURL` referenced by the latest MPD via the delta.

```
647a
                <SegmentUrl mediarange="64712375-65844316"/>
.
582a
                <SegmentUrl mediarange="34678150-35284727"/>
.
517a
                <SegmentUrl mediarange="17339554-17642841"/>
.
21c
      <DeltaSupport sourceURL="delta2.mpdd" availabilityDuration="120s"/>
.
```

At the next MPD update, '`delta1.mpdd`' would contain the cumulative update for 2 MPD updates.

```
647a
                <SegmentUrl mediarange="64712375-65844316"/>
                <SegmentUrl mediarange="65844317-66966044"/>
.
582a
                <SegmentUrl mediarange="34678150-35284727"/>
                <SegmentUrl mediarange="35284728-35885833"/>
.
517a
                <SegmentUrl mediarange="17339554-17642841"/>
                <SegmentUrl mediarange="17642842-17943394"/>
.
21c
      <DeltaSupport sourceURL="delta3.mpdd" availabilityDuration="120s"/>
.
```

Annex E (normative):
Void

Annex F (normative): OMA DM QoE Management Object

As an alternative to configuring the QoE reporting for each session via MPD, OMA-DM can be used to specify the QoE configuration. If such an OMA-DM QoE configuration has been specified, it shall be evaluated by the client for all subsequent sessions.

For the OMA-DM QoE configuration the parameters are specified according to the following Managed Object (MO), and represents the same information as specified in section 10.4 and 10.5. Version numbering is included for possible extension of the MO.

The Management Object Identifier shall be: `urn:oma:mo:ext-3gpp-pss-dash-qoe:1.0`.

Protocol compatibility: The MO is compatible with OMA Device Management protocol specifications, version 1.2 and upwards, and is defined using the OMA DM Device Description Framework as described in the Enabler Release Definition OMA-ERELD _DM-V1_2 [22].

The nodes and leaf objects as provided in Figure F.1 shall be contained under the 3GPP_PSS_DASH_QOE node if a client supports the feature described in this clause.

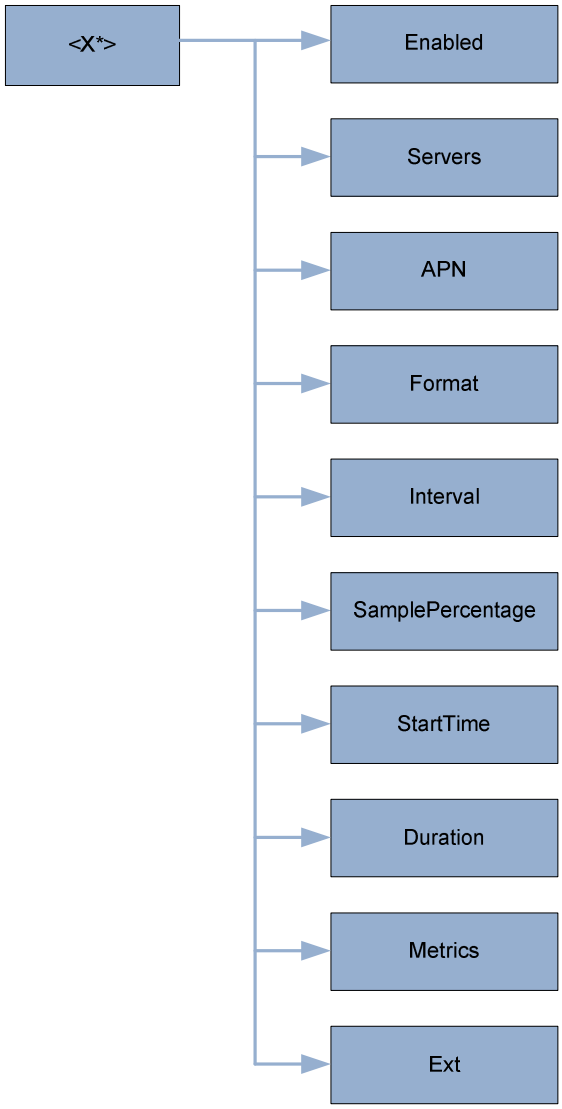

Figure F.1: Nodes and leaf objects

Node: /<X>

This interior node specifies the unique object id of a QoE metrics management object. The purpose of this interior node is to group together the parameters of a single object.

- Occurrence: ZeroOrOne
- Format: node
- Minimum Access Types: Get

The following interior nodes shall be contained if the client supports the QoE Management Object.

/<X>/Enabled

This leaf indicates if QoE reporting is requested by the provider.

- Occurrence: One
- Format: bool
- Minimum Access Types: Get

/<X>/Servers

This leaf contains a space-separated list of servers to which the QoE reports are transmitted. It is URI addresses, e.g. http://qoeserver.operator.com. In case of multiple servers, the client randomly selects one of the servers from the list, with uniform distribution.

- Occurrence: One
- Format: chr
- Minimum Access Types: Get
- Values: URI of the servers to receive the QoE report.

/<X>/APN

This leaf contains the Access Point Name that should be used for establishing the PDP context on which the QoE metric reports will be transmitted. This may be used to ensure that no costs are charged for QoE metrics reporting. If this leaf is not defined then any QoE reporting is done over the default access point.

- Occurrence: ZeroOrOne
- Format: chr
- Minimum Access Types: Get
- Values: The Access Point Name

/<X>/Format

This leaf specifies the format of the report.If this leaf is not defined the QoE reports shall be sent uncompressed.

- Occurrence: ZeroOrOne
- Format: chr
- Minimum Access Types: Get
- Values: 'uncompressed', 'gzip'

/<X>/Interval

This leaf specifies how often QoE reports shall be sent. If this leaf is not defined only one QoE report shall be sent after the complete session.

- Occurrence: ZeroOrOne
- Format: int
- Minimum Access Types: Get
- Values: seconds

/<X>/SamplePercentage

This leaf specifies the percentage of sessions for which QoE metrics shall be reported. The client evaluates a random number at start of each session to determine if reporting shall be done for the specific session. If this leaf is not defined QoE reports are sent for every session.

- Occurrence: ZeroOrOne
- Format: float
- Minimum Access Types: Get
- Values: 0.0-100.0.

/<X>/StartTime

This leaf specifies when collection of QoE metrics shall start. It is specified in seconds and is relative to the start of the session. If this leaf is not defined, the QoE collection shall be done from the start of the session.

- Occurrence: ZeroOrOne
- Format: int
- Minimum Access Types: Get
- Values: seconds

/<X>/Duration

This leaf specifies for how long QoE collection shall be done. It is specified in seconds and is relative to the start time of QoE collection. If this leaf is not defined QoE collection shall be done until the end of the session.

- Occurrence: ZeroOrOne
- Format: int
- Minimum Access Types: Get
- Values: seconds.

/<X>/Metrics

This leaf specifies a list of white-space separated metrics which shall be reported, and follows the same syntax as specified for the "@metrics" attribute in Table 32. If this leaf is not defined no QoE reporting shall be done.

- Occurrence: ZeroOrOne
- Format: chr
- Minimum Access Types: Get
- Values: Metrics as specified in section 10.4.

/<X>/Ext

The Ext node is an interior node where the vendor specific information can be placed (vendor includes application vendor, device vendor etc.). Usually the vendor extension is identified by vendor specific name under the ext node. The tree structure under the vendor identified is not defined and can therefore include one or more un-standardized sub-trees.

- Occurrence: ZeroOrOne
- Format: node
- Minimum Access Types: Get

Annex G (normative): File format extensions for 3GPP DASH support

G.1 Introduction

This clause documents extensions to the ISO base media file format [11] for the support of 3GPP DASH. It is expected that these boxes will be integrated in an updated version of ISO/IEC 14496-12 [11].

G.2 Level Assignment Box

G.2.1 Definition

Box Type: `leva"
Container: Movie Extends Box (`mvex")
Mandatory: No
Quantity: Zero or one

Levels specify subsets of the file. Samples mapped to level n may depend on any samples of levels m, where m <= n, and shall not depend on any samples of levels p, where p > n.

Levels cannot be specified for the initial movie. When the Level Assignment box is present, it applies to all movie fragments subsequent to the initial movie.

For the context of the Level Assignment box, a fraction is defined to consist of one or more Movie Fragment boxes and the associated Media Data boxes, possibly including only an initial part of the last Media Data Box. Within a fraction, data for each level shall appear contiguously. Data for levels within a fraction shall appear in increasing order of level value. All data in a fraction shall be assigned to levels.

NOTE: In the context of 3G DASH, each subsegment indexed within a Subsegment Index box is a fraction.

The Level Assignment box provides a mapping from features, such as temporal sub-sequences, to levels. A feature can be specified through a track or a sample grouping of a track.

The following assignment_types are defined; `assignment_type` values greater than 4 are reserved, while the semantics for the other values are specified as follows.

- 0: sample groups are used to specify levels, i.e. i.e. samples mapped to different sample group description indexes of a particular sample grouping lie in different levels within the identified track; other tracks are not affected and must have all their data in precisely one level;
- 1: as for assignment_type 0 except assignment is by a parameterized sample group;
- 2, 3: level assignment is by track (see the Subsegment Index Box for the difference in processing of these levels)

The sequence of assignment_types is restricted to be a set of zero or more of type 2 or 3, followed by zero or more of exactly one type.

G.2.2 Syntax

```
aligned(8) class LevelAssignmentBox extends FullBox("leva", 0, 0) { unsigned int(8) level_count;
    for (j=1; j <= level_count; j++) {
        unsigned int(32)    track_id;
        unsigned int(1) padding_flag;
        unsigned int(7) assignment_type;
        if (assignment_type == 0)
            unsigned int(32)    grouping_type;
        else if (assignment_type == 1) {
```

```
            unsigned int(32)    grouping_type;
            unsigned int(32)    grouping_type_parameter;
        }
        else if (assignment_type == 2) {} // no further syntax elements needed
        else if (assignment_type == 3) {} // no further syntax elements needed  }
}
```

G.2.3 Semantics

`level_count` specifies the number of levels each fraction is grouped into. `level_count` shall be greater than or equal to 2.

`track_id` for loop entry j specifies the track identifier of the track assigned to level j.

`padding_flag` equal to 1 indicates that a conforming fraction can be formed by concatenating any positive integer number of levels within a fraction and padding the last Media Data box by zero bytes up to the full size that is indicated in the header of the last Media Data box. The semantics of `padding_flag` equal to 0 are unspecified.

`assignment_type` indicates the mechanism used to specify the assignment to a level. `assignment_type` values greater than 3 are reserved, while the semantics for the other values are specified as follows.

`grouping_type` and `grouping_type_parameter`, if present, specify the sample grouping used to map sample group description entries in the Sample Group Description box to levels. Level n contains the samples that are mapped to the sample group description entry having index n in the Sample Group Description box having the same values of `grouping_type` and `grouping_type_parameter`, if present, as those provided in this box.

G.3 Subsegment Index Box

G.3.1 Definition

Box Type: `ssix"
Container: File
Mandatory: No
Quantity: Zero or more

The Subsegment Index box ('ssix') provides a mapping from levels (as specified by the Level Assignment box) to byte ranges of the indexed subsegment. In other words, this box provides a compact index for how the data in is ordered according to levels into partial sub-segments. It enables a client to easily access data for partial subsegments by downloading ranges of data in the subsegment.

Each byte in the subsegment shall be assigned to a level. If the range is not associated with any information in the level assignment, then any level that is not included in the level assignment may be used. Each level shall be assigned to exactly one partial sub-segment, i.e. byte ranges for one level shall be contiguous.

Samples of a partial subsegment may depend on any samples of preceding partial subsegments in the same subsegment, but not the other way around. For example, each partial subsegment contains samples having an identical temporal level and partial subsegments appear in increasing temporal level order within the subsegment.

There may be 0 or 1 Subsegment Index boxes per each Segment Index box that does not refer to other Segment Index boxes, i.e. that only indexes subsegments but no segment indexes. A Subsegment Index box, if any, shall be the next box after the associated Segment Index box. A Subsegment Index box documents the subsegment that is indicated in the immediately preceding Segment Index box.

When a partial segment is accessed in this way, for all `assignment_types` other than 3, the final Media Data box may be incomplete, that is, less data is accessed than the length indication of the Media Data Box indicates is present. The length of the Media Data box may need adjusting, or padding used. The padding_flag in the Level Assignment Box indicates whether this missing data can be replaced by zeros. If not, the sample data for samples assigned to levels that are not accessed is not present, and care should be taken not to attempt to process such samples.

NOTE: `assignment_type` equal to 3 may be used, for example, when audio and video movie fragments (including the respective Media Data boxes) are interleaved. The first level can be specified to contain the audio movie fragments (including the respective Media Data boxes), whereas the second level can be specified to contain both audio and video movie fragments (including all Media Data boxes).

G.3.2 Syntax

```
aligned(8) class SubsegmentIndexBox extends FullBox("ssix", 0, 0) {
    unsigned int(32)    subsegment_count;


    for( i=1; i <= subsegment_count; i++)
      unsigned int(8)   ranges_count;
        for ( j=1; j <= ranges_count; j++) {
            unsigned int(8) level;
            unsigned int(24) accumulated_level_size;
      }
}
```

G.3.3 Semantics

`subsegment_count` is a positive integer specifying the number of subsegments for which partial subsegment information is specified in this box. `subsegment_count` shall be equal `reference_count` (i.e. the number of movie fragment references) in the immediately preceding Segment Index box.

`ranges_count` specifies the number of partial subsegment levels the media data is grouped into. This value shall be greater than or equal to 2.

`range_size` indicates the size of the partial subsegment.

`level` specifies the level to which this partial subsegment is assigned to.

G.4 Temporal level sample grouping

G.4.1 Definition

Many video codecs support temporal scalability where it is possible to extract one or more subsets of frames that can be independently decoded. A simple case is the extraction of I frames for a bitstream with a regular I-frame interval, e.g. IPPPIPPP…, where every 4th picture is an I frame. Also subsets of these I frames can be extracted for even lower frame rates. More elaborate situations with several temporal levels can be constructed using hierarchical B or P frames.

The Temporal Level sample grouping ('tele') provides a codec-independent sample grouping that can be used to group samples (access units) in a track (and potential track fragments) according to temporal level, where samples of one temporal level have no coding dependencies on samples of higher temporal levels. The temporal level equals the sample group description index (taking values 1, 2, 3, etc). The bitstream containing only the access units of from the first temporal level to a higher temporal level remains conforming to the coding standard.

A grouping according to temporal level facilitates easy extraction of temporal subsequences, for instance using the Subsegment Index box in clause G.3.

G.4.2 Syntax

```
class TemporalLevelEntry() extends SampleGroupDescriptionEntry('tele')
{
    bit(1)  level_independently_decodable;
    bit(7)  reserved=0;
}
```

G.4.3 Semantics

The temporal level of samples in a sample group equals to the sample group description index.

`level_independently_decodable` is a flag. 1 indicates that all samples of this level have no coding dependencies on samples of other levels. 0 indicates that no information is provided.

G.5 Producer reference box

G.5.1 Definition

Box Type: `` `prft" ``
Container: File
Mandatory: No
Quantity: Zero or more

The producer reference time box supplies relative wall-clock times at which movie fragments, or files containing movie fragments (such as segments) were produced. When these files are both produced and consumed in real time, this can provide clients with information to enable them to synchronize consumption with the production and thus avoid buffer overflow or underflow.

This box is related to the next movie fragment box that follows it in bitstream order. It must follow any segment type or segment index box (if any) in the segment, and occur before the following movie fragment box (to which it refers). If a segment file contains any producer reference time boxes, then the first of them shall occur before the first movie fragment box in that segment.

The box contains a time value measured on a clock which increments at the same rate as a UTC-synchronized NTP clock, using NTP format. This is associated with a media time for one of the tracks in the movie fragment. That media time should be in the range of times in that track in the associated movie fragment.

G.5.2 Syntax

```
aligned(8) class ProducerReferenceTimeBox extends FullBox("srft", version, 0) {
    unsigned int(32) reference_track_ID;
    unsigned int(64) ntp_timestamp;
    if (version==0)
    {
        unsigned int(32)    media_time;
    } else
    {
        unsigned int(64)    media_time;
    }
}
```

G.5.3 Semantics

`reference_track_ID` provides the `track_ID` for the reference track.
`ntp_timestamp` indicates a UTC time in NTP format corresponding to `decoding_time`.
`media_time` corresponds to the same time as `ntp_timestamp`, but in the time units used for the reference track, and is measured on this media clock as the media is produced. Note that in most cases this timestamp will not be equal to the timestamp of the first sample of the adjacent segment of the reference track, but it is recommended it be in the range of the segment containing this producer reference time box.

G.6 Stream Access Points

G.6.1 Introduction

This Annex defines a Stream Access Point (SAP) and specifies six types of SAPs.

A Stream Access Point (SAP) enables random access into a container of media stream(s). A container may contain more than one media stream, each being an encoded version of continuous media of certain media type. A SAP is a

position in a container enabling playback of an identified media stream to be started using only (a) the information contained in the container starting from that position onwards, and (b) possible initialization data from other part(s) of the container, or externally available. Derived specifications should specify if initialization data is needed to access the container at a SAP, and how the initialization data can be accessed.

G.6.2 SAP properties

For each SAP the properties, I_{SAP}, T_{SAP}, I_{SAU}, T_{DEC}, T_{EPT}, and T_{PTF} are identified and defined as:

- T_{SAP} is the earliest presentation time of any access unit of the media stream such that all access units of the media stream with presentation time greater than or equal to T_{SAP} can be correctly decoded using data in the Bitstream starting at I_{SAP} and no data before I_{SAP}.
- I_{SAP} is the greatest position in the Bitstream such that all access units of the media stream with presentation time greater than or equal to T_{SAP} can be correctly decoded using Bitstream data starting at I_{SAP} and no data before I_{SAP}.
- I_{SAU} is the starting position in the Bitstream of the latest access unit in decoding order within the media stream such that all access units of the media stream with presentation time greater than or equal to T_{SAP} can be correctly decoded using this latest access unit and access units following in decoding order and no access units earlier in decoding order.

NOTE I_{SAU} is always greater than or equal to I_{SAP}.

- T_{DEC} is the earliest presentation time of any access unit of the media stream that can be correctly decoded using data in the Bitstream starting at I_{SAU} and no data before I_{SAU}.
- T_{EPT} is the earliest presentation time of any access unit of the media stream starting at I_{SAU} in the Bitstream.
- T_{PTF} is the presentation time of the first access unit of the media stream in decoding order in the Bitstream starting at I_{SAU}.

G.6.3 SAP types

Six types of SAPs are defined with properties as follows:

- Type 1: $T_{EPT} = T_{DEC} = T_{SAP} = T_{PTF}$
- Type 2: $T_{EPT} = T_{DEC} = T_{SAP} < T_{PTF}$
- Type 3: $T_{EPT} < T_{DEC} = T_{SAP} <= T_{PTF}$
- Type 4: $T_{EPT} <= T_{PTF} < T_{DEC} = T_{SAP}$
- Type 5: $T_{EPT} = T_{DEC} < T_{SAP}$
- Type 6: $T_{EPT} < T_{DEC} < T_{SAP}$

NOTE The type of SAP is dependent only on which Access Units are correctly decodable and their arrangement in presentation order. The types informally correspond with some common terms:

- Type 1 corresponds to what is known in some coding schemes as a 'Closed GoP random access point' (in which all access units, in decoding order, starting from I_{SAP} can be correctly decoded, resulting in a continuous time sequence of correctly decoded access units with no gaps) and in addition the access unit in decoding order is also the first access unit in presentation order.
- Type 2 corresponds to what is know in some coding schemes as a 'Closed GoP random access point', for which the first access unit in decoding order in the media stream starting from I_{SAU} is not the first access unit in presentation order.

- Type 3 corresponds to what is known in some coding schemes as an 'Open GoP random access point', in which there are some access units in decoding order following I_{SAU} that cannot be correctly decoded and have presentation times less than T_{SAP}.

- Type 4 corresponds to what is known in some coding schemes as an "Gradual Decoding Refresh (GDR) random access point', in which there are some access units in decoding order starting from and following I_{SAU} that cannot be correctly decoded and have presentation times less than T_{SAP}.

- Type 5 corresponds to the case for which there is at least one access unit in decoding order starting from I_{SAP} that cannot be correctly decoded and has presentation time greater than T_{DEC} and where T_{DEC} is the earliest presentation time of any access unit starting from I_{SAU}.

- Type 6 corresponds to the case for which there is at least one access unit in decoding order starting from I_{SAP} that cannot be correctly decoded and has presentation time greater than T_{DEC} and where T_{DEC} is not the earliest presentation time of any access unit starting from I_{SAU}.

Annex H (normative): MIME Type Registration for MPD

H.1 MPD MIME Type

H.1.1 Introduction

The MIME type of the MPD is registered and available at the registry at http://www.iana.org/assignments/media-types/application. For formal registration, refer to ISO/IEC 23009-1 [37], Annex C.

H.1.2 Void

H.1.3 Void

H.2 MPD Delta MIME Type

H.2.1 Introduction

This Annex provides the formal MIME type registration for the MPD Delta. It is referenced from the registry at http://www.iana.org/.

H.2.2 MIME Type and Subtype

The MIME Type and Subtype are defined as follows:

Media Type Name: application

Subtype name: Standards Tree - dashdelta

Required parameters: none

Optional parameters: none

Encoding considerations: 8-bit text

Security considerations:
A Media Presentation Description (MPD) Delta contains text changes to an MPD. An MPD Delta is used together with a first MPD to construct a second MPD. As such, any security considerations for an MPD may also be applicable to an MPD Delta. A MIME type handler would not launch a service with only an MPD Delta.
Further to this, as an MPD Delta performs editing operations on an MPD there are risks that deliberately malformed editing operations could cause security issues.

Interoperability considerations:

Published specification: 3GPP TS 26.247.

Applications which use this media type:
various including but not limited to On-Demand Streaming over the Internet, Live Streaming over the Internet, Internet Video, Internet Radio

Additional information:

1. Magic number(s) : none
2. File extension(s) : mpdd
3. Macintosh file type code : none
4. Object Identifiers: none

Person to contact for further information:

1. Name : David Furbeck
2. Email : dfurbeck@blackberry.com

Intended usage : Common

Author/Change controller : 3GPP TSG SA WG4.

Annex I (informative): Signalling of DASH AVP values for QoS handling in the PCC

The PCC architecture is defined in TS 23.203 [31] and provides the Rx reference point, which enables the application layer to authorize a specific usage. In this architecture the DASH HTTP streaming server or any other function in the HTTP streaming path (e.g. an HTTP proxy) can act as Application Function and interact with the PCRF via the Rx reference point for QoS control. It is assumed here that the AF has knowledge of the application type and of the MPD.

The relevant AVPs are the ones enabling the PCRF to establish bearers with correct characteristics for DASH users. The AVPs are defined in TS 29.214 [33]. The further PCRF mapping from AVP to IP QoS parameter mapping is defined in TS 29.213 [32]

Table I.1: Example mapping of MPD parameters to Rx AVPs for 3GP-DASH (PSS)

AVP	Value	Comment
AF-Application-Identifier	'DASH'	Allows to signal the DASH based application hence giving the opportunity to enforce application specific policies
Max-Requested-Bandwidth-DL (NOTE 1)	B1	`B1 = sum of all` **`MPD`**`@maxBandwidth` (see clause 8.4.3.3) of all media components simultaneously (not mutually exclusive) selectable by the DASH client plus HTTP/TCP/IP overhead and TCP messages for flow control. If this attribute is not present then `B1 = sum of` **`MPD`**`@bandwidth` attributes of all media components of the available media presentation corresponding to representations or subrepresentations with highest bandwidth simultaneously selectable (not mutually exclusive) by the DASH client plus HTTP/TCP/IP overhead and TCP messages for flow control. Note: the mapping rules to derive the TCP message flow control bandwidth are FFS.
Max-Requested-Bandwidth-UL (NOTE 1)	FFS	For Further Study. If included, should be greater than or equal to Min-Requested-Bandwidth-UL
Min-Requested-Bandwidth-DL (NOTE 1)	B2	`B2 = sum of all` **`MPD`**`@minBandwidth` (see clause 8.4.3.3) of all media components simultaneously (not mutually exclusive) selectable by the DASH client plus HTTP/TCP/IP overhead and TCP messages for flow control. If this attribute is not present then `B2 = sum of` **`MPD`**`@bandwidth` attributes of all media components of the available media presentation corresponding to representations or subrepresentations with lowest bandwidth simultaneously (not mutually exclusive) selectable by the DASH client plus HTTP/TCP/IP overhead and TCP messages for flow control. Note: the mapping rules to derive the TCP message flow control bandwidth are FFS.
Min-Requested-Bandwidth-UL (NOTE 1)	FFS	For Further Study. Enough bitrate to cover TCP and HTTP GET requests.
Flow-Description AVP (NOTE 1)	IP addresses and ports	

NOTE 1: AVPs provided within the Media-Component-Description AVP, except Flow-Description AVP that is included within the Media-Sub-Component AVP. Omitted AVPs are not relevant for this functionality

Annex J (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
2011-06	52	SP-110305			Version 10.0.0 approved at TSG SA#52		10.0.0
2011-11	54	SP-110794	0003		Alignment with MPEG DASH	10.0.0	10.1.0
2011-11	54	SP-110794	0005	3	QoE Updates for Correction, Clarification and MPEG DASH Alignment	10.0.0	10.1.0
2011-11	54	SP-110794	0006	2	QoS Support for 3GP-DASH Services	10.0.0	10.1.0
2012-06	56	SP-120221	0007	3	Alignment with MPEG DASH	10.1.0	10.2.0
2012-06	56	SP-120221	0010	1	Correction of Table Reference for Change Commands in MPD Deltas	10.1.0	10.2.0
2012-06	56	SP-120221	0012	2	ContentProtection element update to signal version of DRM system	10.1.0	10.2.0
2013-03	59	SP-130015	0021	1	DASH QoE Reporting Schema bug fix	10.2.0	10.3.0
2013-03	59	SP-130015	0023	1	DASH QoE Reporting Example bug fix	10.2.0	10.3.0
2013-06	60	SP-130184	0018	6	General Corrections to DASH	10.3.0	10.4.0
2013-06	60	SP-130184	0025	2	DASH Profiles correction	10.3.0	10.4.0
2013-06	60	SP-130184	0027	1	Correction of Cardinality of DeltaSupport element in Table 8-5	10.3.0	10.4.0
2013-06	60	SP-130184	0029	2	Correction of MPD Delta MIME type information	10.3.0	10.4.0

History

Document history		
V10.0.0	June 2011	Publication
V10.1.0	January 2012	Publication
V10.2.0	July 2012	Publication
V10.3.0	April 2013	Publication
V10.4.0	July 2013	Publication