ISO/IEC JTC 1/SC 29

Date: 2011-01-28

ISO/IEC FCD 23001-6

ISO/IEC JTC 1/SC 29/WG 11

Secretariat:

Information technology — MPEG systems technologies — Part 6: Dynamic adaptive streaming over HTTP (DASH)

Élément introductif — Élément central — Partie 6: Titre de la partie

Document type: International Standard
Document subtype:
Document stage: (40) Enquiry
Document language: E

STD Version 2.1c2

Contents

Page

Foreword

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

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

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

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 23001-6 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 23001 consists of the following parts, under the general title *Information technology — MPEG systems technologies*:

— *Part 1: Binary MPEG format for XML*

— *Part 2: Fragment Request Units*

— *Part 3: XML IPMP messages*

— *Part 4: Codec configuration representation*

— *Part 5: Bitstream Syntax Description Language (BSDL)*

— *Part 6: Dynamic adaptive streaming over HTTP (DASH)*

Introduction

This part of ISO/IEC 23001 specifies formats for adaptive streaming delivery of MPEG media over HTTP. This International Standard is applicable to streaming services over the Internet.

Information technology — MPEG systems technologies — Part 6: Dynamic adaptive streaming over HTTP (DASH)

1 Scope

This part of ISO/IEC 23001 specifies formats for adaptive streaming delivery of MPEG media over HTTP. This International Standard is applicable to streaming services over the Internet.

2 Normative references

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

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

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

ISO/IEC 14496-10, *Information* technology – *Coding of audio-visual objects – Part 10: Advanced Video Coding*

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

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

IETF RFC 2141, *URN Syntax*, May 1997.

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

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

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

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

IETF RFC 4281, *The Codecs Parameter for 'Bucket' Media Types*, November 2005.

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

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

3 Definitions

For the purposes of this International Standard, the following terms and definitions apply.

3.1 Terms and definitions

3.1.1
complementary representation
representation in a set of representations which have inter-representation dependencies and which when combined result in a single representation for decoding and presentation

3.1.2
continuous media
media with an inherent notion of time, for example, speech, audio, video, timed text or timed metadata

3.1.3
HTTP-URL
URL with a fixed scheme of “http://” or “https://”

3.1.4
media component
an encoded version of one individual media type such as audio, video or timed text with specific attributes, e.g. bandwidth, language, or resolution

3.1.5
media content
set of media components, (e.g. audio, video, timed text) that have a common timeline as well as relationships on how they may be presented (for example individually, jointly, or mutually exclusive) with an example being a program or a movie

3.1.6
media presentation
structured collection of data that establishes a bounded or unbounded presentation of media content composed of components of continuous media

3.1.7
media presentation description
formalized description for a media presentation

3.1.8
period
subset of the media presentation where a contiguous sequence of all periods constitutes the media presentation.

3.1.9
representation
structured collection of one or more media components

3.1.10
representation access point
position in a media segment that is identified as being a position for which it is possible to start playback using only the information contained in the media segment from that position onwards (preceded by initialising with the initialisation segment, if any). It consists of a byte index, I_{RAP}, and a presentation time, T_{RAP}, related as follows

— T_{RAP} is the earliest presentation time such that all access units with presentation time greater than or equal to T_{RAP} can be correctly decoded using stream data starting at I_{RAP} and no stream data before I_{RAP}.

— I_{RAP} is the greatest byte index in the stream such that all access units with presentation time greater than or equal to T_{RAP} can be correctly decoded using stream data starting at I_{RAP} and no stream data before I_{RAP}.

Representation access points may coincide with random access points in certain media streams.

3.1.11
segment
a resource that can be identified by an HTTP-URL and possibly a byte-range, included in the MPD, which when this entire resource thus identified is requested through HTTP/1.1 GET method (or partial GET with the indicated byte range) as defined in RFC 2616 the segment is the entity body of the request response.

3.1.12
subsegment
smallest unit within segments which may be indexed at the segment level.

3.2 Symbols and abbreviated terms

For the purpose of this document, the symbols and abbreviated terms given in the following apply:

AVC advanced video coding

CAT conditional access table

DASH dynamic adaptive streaming over HTTP

ECM entitlement control message

HTTP hypertext transfer protocol

IDR instantaneous decoding refresh

MPD media presentation description

MVC multi-view video coding

OP observation point

PAT program association table

PCR program clock reference

PES packetized elementary stream

PID packet identifier

PMT program map table

PSI program specific information

PTS presentation time stamp

QM quality metrics

RAP representation access point

SVC scalable video coding

TCP transmission control protocol

TLS transport layer security

TS transport stream

URI uniform resource identifier

URL uniform resource locator

URN uniform resource name

UUID universally unique identifier

UTC coordinated universal time

XML extensible mark-up language

3.3 Conventions

The following naming conventions apply in this document:

— 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`**.

— 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`.

— Variables defined in the context of this 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. Period, Group, Representation, Segment, etc.

4 Introduction

4.1 System description

Dynamic Adaptive Streaming over HTTP (DASH) specifies formats that enable delivery of media content from standard HTTP servers to HTTP clients and enable caching of content by standard HTTP caches.

This specification primarily defines two formats:

— 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. In the context of this specification, the resource identifiers are exclusively HTTP-URLs, i.e. URLs with a fixed scheme as defined in RFC 3986 of "http://" or "https://". However, this specification additionally enables the restriction of these URLs by a byte range attribute.

— The Segment formats specify the formats of the entity body of the request response when issuing a HTTP GET request or a partial HTTP GET with the indicated byte range through HTTP/1.1 as defined in RFC 2616 to a resource identified in the MPD.

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 the specification exclusively HTTP/1.1. We refer to such a client as a DASH Client in the remainder of this document. However, this standard does not provide a normative definition for such a client.

Figure 1 shows a possible deployment architecture in which the formats defined in this standard may be used. This standard deals with the definition of formats that are accessible on the interface to the DASH client. Any other formats or interfaces are not in scope of this standard. In the considered deployment scenario, it is assumed that the DASH Client has access to an MPD. The MPD provides sufficient information for the DASH Client to provide a streaming service to the user by downloading Segments from an HTTP server and demultiplexing, decoding and rendering the included media data appropriately.

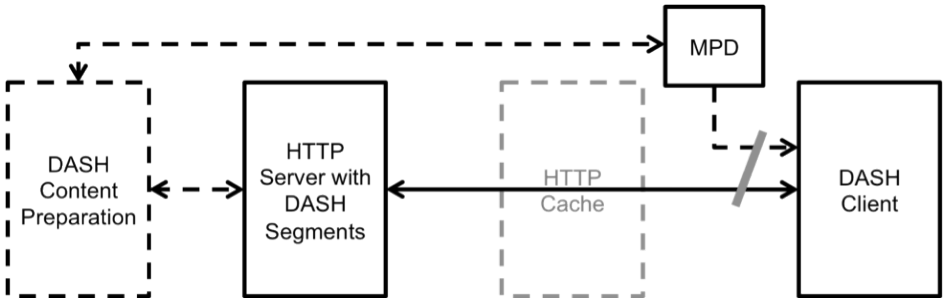

Figure 1 — Possible Deployment Architecture for DASH

4.2 DASH Client Model

The design of the formats defined in this standard is based on a client model as shown in Figure 2. The figure illustrates the logical components of a DASH client. In this figure the DASH Access Client 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 Client consists of media in MPEG container formats (ISO Base Media File Format ISO/IEC 14496-12 or MPEG-2 Transport Stream ISO/IEC 13818-2), or parts thereof, together with timing information that maps the internal timing of the media to the time line of the Media Presentation. In 6.1 and 7.1 of this Part of ISO/IEC 23001, guidance on enabling the use of this standard with other container formats is provided.

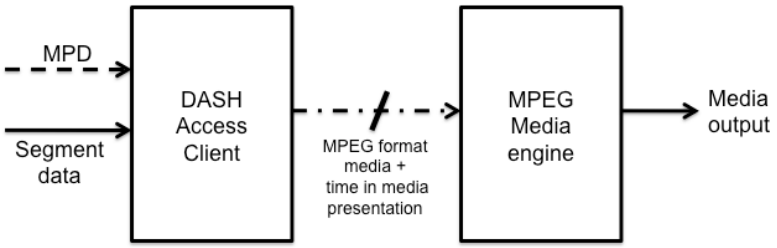

Figure 2 — DASH Client Model

4.3 Protocols

This standard may be deployed in a system according to Figure 1 for which

— The DASH Client complies with a *client* as specified in RFC 2616,

— The HTTP Server hosting the DASH Segments complies with a *server* as specified in RFC 2616,

DASH Clients typically use the HTTP GET method or the HTTP partial GET method, as specified in RFC 2616, Subclause 9.3, to access Segments or parts thereof.

Transport security in HTTP-based delivery may be achieved using the HTTP over TLS as specified in RFC 2818.

5 Media Presentation

5.1 General

A Media Presentation is a structured collection of data that is accessible to a DASH Client to provide a streaming service to the user.

A Media Presentation is described by a MPD including any possible updates of the MPD. The MPD is defined in 5.2 and the update mechanisms in 5.8. Assembly of a fragmented MPD is defined in 5.3. The data model that constitutes a media presentation is defined in 5.4 and some additional elements in the MPD that describe the content are provided in 5.5, 5.6, and 5.7.

5.2 Media Presentation Description

5.2.1 General

The Media Presentation Description (MPD) is a document that contains metadata required by a DASH Client to construct appropriate HTTP-URLs to access Segments and to provide the streaming service to the user. HTTP-URLs may be absolute or relative. If relative then reference resolution as defined in 5.2.3 shall be applied. Handling of alternative base URLs is addressed in 5.2.4.

The MPD is an XML-document that is formatted according to the XML schema provided in 5.2.2.

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 5.2.2 and that complies with this standard.

The MPD shall contain exactly one `MPD` element as defined in 5.4.

The MIME type of the MPD is defined in Annex D.

The delivery of the MPD is not in scope of this standard. However, if the MPD is delivered over HTTP, then the MPD may be transfer encoded for transport, as described in RFC 2616.

EXAMPLE the generic GZip algorithm as defined in RFC 1952, Clause 9 may be used for transfer-encoding.

5.2.2 Schema

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

> NOTE the namespace identifier is for this draft version of the specification ("urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"). The final version will use a different namespace identifier to align with 3GPP However, changes from the draft to the final version are not expected to change the information model, even if the schema changes, and trial implementation and comment on the schema presented here are both encouraged.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"
     attributeFormDefault="unqualified"
     elementFormDefault="qualified"
     xmlns:xs="http://www.w3.org/2001/XMLSchema"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     xmlns="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011">
```

```
    <xs:import namespace="http://www.w3.org/1999/xlink" schemaLocation="xlink.xsd"/>

        <xs:annotation>
         <xs:appinfo>Media Presentation Description</xs:appinfo>
         <xs:documentation xml:lang="en">
              This Schema defines Media Presentation Description for MPEG DASH.
              the namespace identifier is for this draft version of the specification.
             The final version will use a different namespace identifier to align with 3GPP.
             However, changes from the draft to the final version are not expected to change
             the information model, even if the schema changes, and trial implementation and
             comment on the schema presented here are both encouraged.
         </xs:documentation>
        </xs:annotation>

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


        <!-- the remaining types, elements attributes are defined in the below -->
        ...

</xs:schema>
```

5.2.3 Reference Resolution

URLs at each level of the MPD are resolved 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 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 RFC 3986 then the MPD should not be interpreted.

In addition to the document level, base URL information may present on the following levels:

— On MPD level in **`MPD.BaseURL`** element, see 5.4.1.

— On Period level in **`Period.SegmentInfoDefault.BaseURL`** element. For details refer to 5.4.4.

— On Group level in **`Group.SegmentInfoDefault.BaseURL`** element. For details refer to 5.4.4.

— On Representation level in **`SegmentInfo.BaseURL`**. For details refer to 5.4.4.

5.2.4 Alternative Base URLs

If alternative base URLs are provided through the **`BaseURL`** element at any level, this means that the identical segments are accessible at multiple locations. In the absence of other criteria, the DASH Client may use the first base URL 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.

5.3 MPD Assembly

5.3.1 Introduction

This Subclause defines a mechanism for referencing a *remote DASH element* from within a local MPD. Therefore, a subset of W3C XLINK simple links is defined as follows by

— restricted syntax and semantics in 5.3.2, and

— the processing model in 5.3.3.

5.3.2 Syntax and semantics

Table 1 provides the XLINK attributes that are used in this standard.

Table 1 — XLINK attributes used in this standard

Attribute	*Comments and Usage*
`@xlink:type`	Identifies the type of W3C XLINK being used. In the context of standard, all references shall be W3C XLINK simple links. As the attribute `@xlink:type` is optional with fixed setting `@xlink:type="simple"` the type of the link shall not be specified.
`@xlink:href`	Identifies the remote Element by URI as defines in IETF RFC 3986. In the context of this standard, 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 standard, as the attribute `@xlink:show` is optional with fixed setting `@xlink:show="embed"` the show attribute of the link should not be specified. 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 5.3.3.
`@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 MPD 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 is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<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>
```

5.3.3 Processing

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

1) URI references to elements that cannot be resolved shall be treated as invalid references.

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

3) URI references that directly or indirectly reference themselves are treated as invalid circular references.

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

1) Attributes shall be added to the element of the MPD that contains `@xlink:href` merging with existing attributes. If the same attributes are present in both MPD and remote DASH 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 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 shall be included in a remote DASH element.

5.4 Hierarchical Data Model

5.4.1 Introduction

5.4.1.1 Overview

A Media Presentation is described in the `MPD` element that is contained in an MPD document formatted as defined in 5.2.

A Media Presentation consists of

— A sequence of one or more *Periods* described in 5.4.2.

— Each Period contains one or more *Groups* described in 5.4.3.3.

— Each Group contains one or more *Representations* from the same media content. Representations are described in 5.4.3.4.

— Each Representation consists of one or more Segments. Segment Information is introduced in 5.4.4. Segments contain media data and/or metadata to access, decode and present the included media content.

This **MPD** element provides descriptive information that enables a client to choose Representations. For doing so, it provides descriptions of the Representations that may also be deduced by inspecting this Representation if available partially or in its entirety to the client. However, actual playback of the Representations is not controlled by the MPD information. Playback is controlled by the media engine operating on the media data in the usual way.

The Media Presentation timeline is defined by the concatenation of the timeline of each Period.

NOTE The playout procedure of the media may need to be adjusted at the end of the preceding Period to match the start time of the new Period as there may be small overlaps or gaps with a Representation at the end of the preceding Period.

The timeline in each Period is common to all Representations.

The summary of the semantics of the attributes and elements within an **MPD** element are provided in Table 2 of 5.4.1.2. The XML-syntax of the **MPD** element is provided in 5.4.1.3.

5.4.1.2 Semantics

Table 2 — Semantics of MPD element

Element or Attribute Name	Use	Description
MPD	1	The root element that carries the Media Presentation Description for a Media Presentation.
@profile	O	A space delimited list of Media Presentation profiles as described in Clause 8
@type	OD default: OnDemand	"OnDemand" or "Live". Indicates the type of the Media Presentation. Currently, on-demand and live types are defined.
@availabilityStartTime	CM Must be present for type="Live"	Gives the earliest availability time (in UTC) for any Segment in the Media Presentation. For @type="Live" this attribute shall be present. In this case it gives the anchor for the computation of the earliest availability time (in UTC) for any Segment in the Media Presentation. For @type="OnDemand" all Segments described in the MPD shall be available. If not present, all Segments described in the MPD shall be available.
@availabilityEndTime	O	Gives the earliest availability end time (in UTC). After this time, the Media Presentation described in this MPD is no longer guaranteed to be available. When not present, the value is unknown.
@mediaPresentationDuration	O	Specifies the duration of the entire Media Presentation. If the attribute is not present, the duration of the Media Presentation is unknown.

@minimumUpdatePeriodMPD	O	Provides the minimum period the containing MPD document is updated. If not present the minimum update period is assumed to be infinite.
@minBufferTime	O	Provides the minimum amount of initially buffered media that is needed to ensure smooth playout provided that each Representation is continuously delivered at or above the value of its @bandwidth attribute. If not present, each **Period** element shall contain the @minBufferTime attribute.
@timeShiftBufferDepth	O	Indicates the duration of the time shifting buffer that is guaranteed to be available for a Media Presentation with type 'Live'. When not present, the value is unknown. This value of the attribute is undefined if the type attribute is equal to 'OnDemand'
ProgramInformation	0…1	Provides descriptive information about the program. For more details refer to the description in 5.5.
Period	1…N	Provides the information of a Period. For more details refer to the description in 5.4.2.
BaseURL	**0…N**	A Base URL that can be used for reference resolution and alternative URL selection. For more details refer to the description in 5.2.3 and 5.2.4.

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 @

5.4.1.3 XML Syntax

```
<!-- MPD Type -->
<xs:complexType name="MPDtype">
    <xs:sequence>
        <xs:element name="ProgramInformation" type="ProgramInformationType" minOccurs="0"/>
        <xs:element name="Period" type="PeriodType" maxOccurs="unbounded"/>
        <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="profiles" type="URIVectorType"/>
    <xs:attribute name="type" type="PresentationType" default="OnDemand"/>
    <xs:attribute name="availabilityStartTime" type="xs:dateTime"/>
    <xs:attribute name="availabilityEndTime" type="xs:dateTime"/>
    <xs:attribute name="mediaPresentationDuration" type="xs:duration"/>
    <xs:attribute name="minimumUpdatePeriodMPD" type="xs:duration"/>
    <xs:attribute name="minBufferTime" type="xs:duration"/>
    <xs:attribute name="timeShiftBufferDepth" 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="OnDemand"/>
        <xs:enumeration value="Live"/>
    </xs:restriction>
</xs:simpleType>

<!-- Supplementary URL to the one given as attribute -->
<xs:complexType name="BaseURLType">
    <xs:simpleContent>
            <xs:extension base="xs:anyURI">
                <xs:anyAttribute namespace="##other" processContents="lax"/>
```

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


        <!-- Type for space delimited list of URIs -->
        <xs:simpleType name="URIVectorType ">
            <xs:list itemType="xs:anyURI"/>
        </xs:simpleType>
```

5.4.2 Period

5.4.2.1 Overview

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

For live services, the `@start` attribute of the first Period shall be present. For on-demand services the `@start` attribute of the first Period, if present, shall be zero. If not present a default value of zero is assumed.

Each Period has a conceptual start time *PeriodStart* in the Media Presentation that is relative to the start of the first Period. Period elements shall be physically ordered in the MPD file in increasing order of their *PeriodStart* time.

The *PeriodStart* time is determined as follows:

— If the attribute `@start` is present in the **`Period`**, then *PeriodStart* is identical to the value of this attribute and it overrides any possibly present prior `@duration` attribute information.

— If the `@start` attribute is absent, the previous **`Period`** element shall contain the `@duration` attribute. Then the start time of the new Period *PeriodStart* is determined as the sum of the start time of the previous Period *PeriodStart* and the value of the attribute `@duration` of the previous Period.

For live services, the sum of the *PeriodStart* value of the Period and the value of the attribute **`MPD`**`@availabilityStartTime` specifies the availability start time of the Period in UTC, in particular the first Media Segment of each Representation in this Period. It also serves as an anchor for the deriving the availability start time of any other Segment in the Period.

Each Period extends until the *PeriodStart* of the next Period, or until the end of the Media Presentation in the case of the last Period.

PeriodStart times reflect the actual UTC time that should elapse after playing the media of all prior Periods in this Media Presentation.

The semantics of the attributes and elements within a **`Period`** element are provided in Table 3 of 5.4.2.2. The XML-syntax of the Period type is provided in 5.4.2.3.

5.4.2.2 Semantics

Table 3 — Semantics of `Period` Element

Element or Attribute Name	Use	Description
`Period`	1…N	Provides the information of a Period.
`@xlink:href`	O	provides a reference to an external **`Period`** element

@xlink:actuate	O default: onRequest	provides 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	O	Provides a unique identifier for this Period within the Media Presentation. The @id of the Period shall remain unchanged over an MPD update.
@start	O	Provides the start time of the Period.
@duration	O	Provides the duration of the Period.
@minBufferTime	O	When present, this attribute overrides for this Period the **MPD**@minBufferTime attribute at MPD level.
@segmentAlignmentFlag	OD Default: false	When set to 'true', indicates that all presentation start and end times of media components of any particular media type are temporally aligned in all Segments, except possibly the last one, across all Representations with the same value of the @duration attribute on Representation level in this Period."
@bitstreamSwitchingFlag	OD Default: false	When this flag is set to 'true', the concatenation of any Initialisation Segment within the same Group in a Period, if present, with all consecutive Media Segments from any Representation within this same Group, starting with the first Media Segment, results in a syntactically valid bitstream (according to the specific bitstream format) that is also semantically correct (i.e. if the concatenation is played, the media content within this Period is correctly presented). This flag shall not be set to 'true' when @segmentAlignmentFlag is set to 'false'. More detailed rules may be defined for specific Initialisation and Media Segment formats.
SegmentInfoDefault	0...1	Provides default Segment information about Segment durations and, optionally, URL construction. For more details see 5.4.4.
Subset	0...N	This element contains a description of a Subset. For more details see 5.4.3.6.
Group	0...N	This element contains a description of a Group. For more details see 5.4.3.3.
Representation	0…N	This element contains a description of a Representation. For more details see 5.4.3.4.

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 @.

5.4.2.3 XML Syntax

```
<!-- Period of a presentation -->
<xs:complexType name="PeriodType">
    <xs:sequence>
        <xs:element name="SegmentInfoDefault" type="SegmentInfoDefaultType" minOccurs="0"/>
        <xs:element name="Representation" type="RepresentationType"
                                    minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="Group" type="GroupType"
                                    minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="Subset" type="SubsetType" 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="start" type="xs:duration"/>
    <xs:attribute name="id" type="xs:string" />
    <xs:attribute name="duration" type="xs:duration"/>
    <xs:attribute name="minBufferTime" type="xs:duration"/>
    <xs:attribute name="segmentAlignmentFlag" type="xs:boolean" default="false"/>
    <xs:attribute name="bitStreamSwitchingFlag" type="xs:boolean" default="false"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

5.4.3 Representation and Grouping

5.4.3.1 Overview

Each Period consists of one or more *Groups*, which each consists of one or more *Representations*.

Groups are described by the **`Group`** element as defined in 5.4.3.3 and are contained in the **`Period`** element.

Representations are described by the **`Representation`** element as defined in 5.4.3.4 contained in the **`Group`** element.

In case no **`Group`** element is present in a Period, Representation elements may be directly contained in a **`Period`** element. In this case a Representation shall be assigned to a Group by the `@group` attribute on both the **`Representation`** and the **`Group`**.

Representations are described in the same Group to signal that they are alternatives to each other and typically contain different encoded versions of the same source material. In this typical case, the ability for seamless switching is a desired feature. 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.

Groups and Representations share some common elements and attributes as collected in 5.4.3.2.

Subsets provide a mechanism to restrict the combination of active Groups. Subsets are described by the **`Subset`** element as defined in 5.4.3.6.

5.4.3.2 Common Group and Representation Attributes and Elements

5.4.3.2.1 Overview

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

The attributes @width, @height, @parx, @pary, @lang, @numberOfChannels, @samplingRate, @mimeType, @group, @startWithRAP, and @frameRate may be present in all three elements. The semantics of these attributes are provided 5.4.3.2.2.

If a **Representation** element is contained in a **Group** element, then an attribute from the above list that is specified in both the **Representation** element and the containing **Group** element shall have the same value in both.

The @group attribute, when provided in a **Group** element, identifies the group.

When provided in a **Representation** element the @group attribute identifies the Group to which the Representation belongs. If not provided for a **Representation** element which is not included within a **Group** element then the Representation shall be assumed to be in group zero.

The semantics of the common attributes and elements are provided in Table 4 in 5.4.3.2.2, the syntax is provided in 5.4.3.2.3.

5.4.3.2.2 Semantics

Table 4 — Common Group, Representation and Sub-Representation Attributes and Elements

Element or Attribute Name	Use	Description
Group, Representation, SubRepresentation		Element
@width	O	Specifies the horizontal visual presentation size of the video media type in an alternative Representation on a square grid determined by the @parx and @pary attributes.
@height	O	Specifies the vertical visual presentation size of the video media type in an alternative Representation, on a square grid determined by the @parx and @pary attributes. This value should be equal to the vertical pixel resolution of the video.
@parx	O	indicates the horizontal size of the encoded video pixels (samples) (in arbitrary units). The default value is 1.
@pary	O	indicates the vertical size of the encoded video pixels (in the same arbitrary units as @parx). The default value is 1.
@frameRate	O	Specifies the output frame rate or the output field rate of the video media type in the representation for progressive or interlaced video, respectively. If the frame or field rate is varying, the value is the average frame or field rate over the entire duration of the representation. In case of a multiview complementary Representation, the value indicates the frame or field rate of a single view.

@lang	O	Declares the language code(s) for this Representation according to IETF RFC 5646. Note, multiple language codes may be declared as a white-space separated list and indicate that the representation may suit a preference for any of the indicated languages. For a full indication of what media is offered under each language, the Initialisation Segment or a Media Segment may have to be accessed.
@numberOfChannels	O	A single value describing the number of audio output channels or a list of available audio channels. For example, @numberOfChannels="5.1 2" for an MPEG Surround Representation
@samplingRate	O	A single value describing the sample rate of the audio stream or a list of sample rates available in the audio stream, e.g. @samplingRate="44100 22050" for an HE-AAC stream with the SBR tool enabled and backwards compatible signaling.
@mimeType	M	Gives the MIME type of the Initialisation Segment, if present; if the Initialisation Segment is not present it provides the MIME type of the first Media Segment. Where applicable, this MIME type shall include the codec parameters for all media types. The codec parameters shall also include the profile and level information where applicable.
@group	O	Specifies the group.
@maximumRAPPeriod	O	Provides the maximum time interval between RAPs in seconds in this Representation. If not present, it is unspecified. The index and the presentation time of any RAP shall either be documented by the Segment Index or is implicitly defined by the @startWithRAP attribute set to 'true'.
@startWithRAP	O	When 'true', indicates that all Segments in the Representation start with a RAP (both in terms of data and in terms of presentation time). The presentation time of the RAP shall either be provided explicitly by the Segment Index or, if the @segmentAlignmentFlag is true, may be inferred from the presentation time of the last sample of the previous segment.
ContentProtection	0 … N	Provides information about the use of content protection for this Representation or Group of Representation. When not present the content is neither encrypted nor DRM protected. When multiple elements are present, then the successful processing of one of the elements is sufficient to access the described Representations.
Accessibility	0 … N	Provides information about Accessibility Information scheme

Rating	0 … N	Provides information Content rating scheme
Viewpoint	0 … N	Provides information Content View Point annotation scheme
MultipleViews	0 … 1	Provides information for video that contains multiple views

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 @.

5.4.3.2.3 XML Syntax

```
    <!-- RepresentationBase type; extended by other Representation-related types -->
    <xs:complexType name="RepresentationBaseType">
        <xs:sequence>
            <xs:element name="ContentProtection" type="ContentDescriptorType"
                                    minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Accessibility" type="ContentDescriptorType"
                                    minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Rating" type="ContentDescriptorType"
                                    minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Viewpoint" type="ContentDescriptorType"
                                    minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="MultipleViews" type="MultipleViewsType" minOccurs="0"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="group" type="xs:unsignedInt"/>
        <xs:attribute name="width" type="xs:unsignedInt"/>
        <xs:attribute name="height" type="xs:unsignedInt"/>
        <xs:attribute name="parx" type="xs:unsignedInt"/>
        <xs:attribute name="pary" type="xs:unsignedInt"/>
        <xs:attribute name="lang" type="LangVectorType"/>
        <xs:attribute name="mimeType" type="xs:string"/>
        <xs:attribute name="startWithRAP" type="xs:boolean"/>
        <xs:attribute name="frameRate" type="xs:double"/>
        <xs:attribute name="maximumRAPPeriod" type="xs:double"/>
        <xs:attribute name="numberOfChannels" type="StringVectorType"/>
        <xs:attribute name="samplingRate" type="StringVectorType"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <!-- Type for space delimited list of strings -->
    <xs:simpleType name="StringVectorType ">
        <xs:list itemType="xs:string"/>
    </xs:simpleType>

  <!-- Type for space delimited list of language codes -->
    <xs:simpleType name="LangVectorType">
        <xs:list itemType="xs:language"/>
    </xs:simpleType>
```

5.4.3.3 Group

5.4.3.3.1 Overview

Groups are described by the **Group** element. This element supports the description of ranges for the @bandwidth, @width, @height and @frameRate attributes defined for the **Representation** element, which provide a summary of all values for all the Representations within this Group. The Representations associated with a **Group** element shall not contain values outside the ranges documented for that Group.

The semantics of the attributes and elements within a **Group** element are provided in Table 5 of 5.4.3.3.2. The XML-syntax of the Period type is provided in 5.4.3.3.3.

5.4.3.3.2 Semantics

Table 5 — Semantics of `Group` element

Element or Attribute Name		Use	Description
`Group`			Group description
	`@xlink:href`	O	reference to external **`Group`** element
	`@xlink:actuate`	OD default: “onRequest”	provides the processing instructions, which can be either "`onLoad`" or "`onRequest`".
	`CommonAttributesElements`	-	Common Representation and Group Attributes and Elements (see 5.4.3.2)
	`@minBandwidth`	O	Minimum bandwidth value in all Representations in this Group.
	`@maxBandwidth`	O	Maximum bandwidth value in all Representations in this Group.
	`@minWidth`	O	Minimum width value in all Representations in this Group.
	`@maxWidth`	O	Maximum width value in all Representations in this Group.
	`@minHeight`	O	Minimum height value in all Representations in this Group.
	`@maxHeight`	O	Maximum height value in all Representations in this Group.
	`@minFrameRate`	O	Minimum frame rate value in all Representations in this Group.
	`@maxFrameRate`	O	Maximum frame rate value in all Representations in this Group.
	`@segmentAlignmentFlag`	O	If given, overrides **`Period`**`@segmentAlignmentFlag` in and specifies the value for all Representations in this Group.
	`@bitStreamSwitchingFlag`	O	If given, overrides **`Period`**`@bitStreamSwitchingFlag` iand specifies the value for all Representations in this Group.
	`@subsegmentAlignment`	OD default: false	Assume the following definitions: — The *Earliest* Presentation Time of a subsegment is defined to be the earliest presentation time of any sample in the reference stream of the subsegment. The Latest Presentation Time of a subsegment is defined to be the Presentation Time of the latest sample in the reference stream of the subsegment that has a Presentation Time less than the Earliest Presentation time of the next subsegment. If the subsegmentAlignment for a Group is set to ‘`true`’,

			all following conditions shall be satisfied: — Each Media Segment shall be indexed (i.e. either it contains a segment index or there is an Index Segment providing an index for the Media Segment) — For every subsegment with `contains_RAP =1`, the T_{RAP} value of the first RAP in the subsegment shall be the earliest presentation time of any sample of the reference stream of the subsegment in which it appears, — For any two representations, A and B, and any k, the Earliest Presentation Time of the kth subsegment of A shall be greater than the Latest Presentation Time of the k-1th subsegment of B.
	`SegmentInfoDefault`	0...1	If present, replaces any possibly present **`Period@SegmentInfoDefault`**. For more details refer to 5.4.4.
	`Representation`	0… N	See 5.4.3.4.

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***

5.4.3.3.3 XML Syntax

```
        <!-- Group to contain information common to a group;
                    extends RepresentationBaseType -->
    <xs:complexType name="GroupType">
        <xs:complexContent>
            <xs:extension base="RepresentationBaseType">
                <xs:sequence>
                    <xs:element name="Representation" type="RepresentationType"
                                                minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="SegmentInfoDefault" type="SegmentInfoDefaultType" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute ref="xlink:href"/>
                <xs:attribute ref="xlink:actuate" default="onRequest"/>
                <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="xs:double"/>
                <xs:attribute name="maxFrameRate" type="xs:double"/>
                <xs:attribute name="subsegmentAlignment" type="xs:boolean" default="false"/>
                <xs:attribute name="segmentAlignmentFlag" type="xs:boolean"/>
                <xs:attribute name="bitStreamSwitchingFlag" type="xs:boolean"/>
                <xs:anyAttribute namespace="##other" processContents="lax"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

5.4.3.4 Representation

5.4.3.4.1 Overview

A Representation is one of the alternative choices of the media content or a subset thereof typically differing by the encoding choice, e.g. by bitrate, resolution, language, codec, etc.

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.

A Representation consists of one or more Segments.

Each Representation either contains an Initialisation Segment or each Media Segment in the Representation is self-initialising.

Each Representation includes one or more media components, where each media component is an encoded version of one individual media type of continuous media such as audio, video, or timed text. Media *components* are time-continuous across boundaries of consecutive Media Segments within one Representation. The timing within each Representation is relative to the *PeriodStart* time or the Period that contains this Representation.

A Representation may contain zero or more Sub-Representations as defined in 5.4.3.5.

The semantics of the attributes and elements within a Representation are provided in Table 6 of 5.4.3.4.2. The XML-syntax of the Representation type is provided in 5.4.3.4.3.

5.4.3.4.2 Semantics

Table 6 —Semantics of `Representation` element

Element or Attribute Name	Use	Description
`Representation`	M	This element contains a description of a Representation.
`@id`	M	Unique identifier for this Representation within the Period. The string shall only contain characters that permit to form a valid HTTP-URL according to RFC 1738.
`@bandwidth`	M	The minimum bandwidth of a hypothetical constant bitrate channel in bits per second (bps) over which the Representation (i.e. the collection of all Segments of a Representation) can be continuously delivered such that a client, after buffering for exactly `@minBufferTime` when accessing a Representation at any RAP can be assured of having enough data for continuous playout.
`@qualityRanking`	O	Provides a quality ranking of the Representation relative to other Representations in the Period. Lower values represent higher quality content. If not present then the ranking is undefined.
`@dependencyId`	O	Indicates a whitespace-separated list of `@id` attributes indicating all complementary Representations the Representation depends on in the decoding process.
`@bitstreamStructureId`	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

		`@bitstreamStructureId` indicates a whitespace-separated list of bitstream structure identifier values. A bitstream formed by concatenating the bitstream of a first representation until a Random Access Point access unit (exclusive) and the bitstream of a second representation (having the same bitstream structure identifier value as for the first representation) starting from the respective Random Access Point access unit (inclusive) conforms to the respective codec specification regardless of the type of the Random Access Point access unit. Furthermore, the decoded pictures have an acceptable quality regardless of type of the Random Access Point access unit used. For AVC, SVC, and MVC bitstreams, a Random Access Point access unit is an Instantaneous Decoding Refresh (IDR) access unit or an access unit for which the recovery_frame_cnt syntax element of the Recovery Point Supplementary Enhancement Information message of AVC can be set to 0. All bitstream structure identifier values for one Group shall differ from those of another Group. NOTE: Indicating multiple bitstream 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 bitstreamStructureId equal to “1”, Representation B would contain bitstreamStructureId equal to “1 2”, and Representation C would contain bitstreamStructureId equal to “2”.
CommonAttributesElements	-	Common Representation and Group Attributes and Elements (see 5.4.3.2)
SubRepresentation	0 … N	Provides information about a sub-representation that is embedded in the containing Representation.
TrickMode	0 … 1	Provides the information for trick mode. It also indicates that the Representation may be used as a trick mode Representation.
SegmentInfo	1	Provides Segment access 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***		

5.4.3.4.3 XML Syntax

```
<!-- Representation of the presentation content for a specific Period;
                    extends RepresentationBaseType -->
<xs:complexType name="RepresentationType">
    <xs:complexContent>
```

```
            <xs:extension base="RepresentationBaseType">
                <xs:sequence>
                    <xs:element name="SubRepresentation" type="SubRepresentationType"
                                                minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="SegmentInfo" type="SegmentInfoType"/>
                    <xs:element name="TrickMode" type="TrickModeType" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="id" type="xs:string" use="required"/>
                <xs:attribute name="bandwidth" type="xs:unsignedInt" use="required"/>
                <xs:attribute name="qualityRanking" type="xs:unsignedInt"/>
                <xs:attribute name="dependencyId" type="StringVectorType"/>
                <xs:attribute name="bitstreamStructureId" type="StringVectorType"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

5.4.3.5 Sub-Representation

5.4.3.5.1 Overview

Sub-Representations are embedded in regular Representations and are described by the **`SubRepresentation`** element. Sub-Representations are not offered as regular Representations, but typically provide the ability for accessing a lower quality version of the Representation in which they are contained. Sub-Representations for example allow extracting the audio track in a multiplexed Representation or may allow for efficient fast-forward operations if provided with lower frame rate. The media data for Sub-Representations is ordered into multiple levels, each providing an enhancement compared to the lower levels. Sub-Representations are essentially described by the same attributes as Representation.

If Sub-Representations are offered, then the initialisation segment and/or the media segments shall provide sufficient information such that the data can be easily accessed through HTTP partial GET requests.

The semantics of the attributes and elements within a Representation are provided in Table 7 of 5.4.3.5.2. The XML-syntax of the Representation type is provided in 5.4.3.5.3.

5.4.3.5.2 Semantics

Table 7 —Semantics of `SubRepresentation` Element

Element or Attribute Name	Use	Description
`SubRepresentation`	M	This element describes a Sub-Representation.
`@level`	M	Specifies the sub-representation level.
`@bandwidth`	M	Identical to the `@bandwidth` definition in Representation, but applied to this Sub-Representation
`@trickModeApr`	O	Specifies the maximum playout rate as a multiple of the regular playout rate, which this sub-representation supports with the same decoder profile and level requirements as the normal playout rate of the sub-representation.
`CommonAttributesElements`	-	Common Representation and Representation Group Attributes and Elements (see 5.4.3.2)

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, List of elements and attributes is in ***italics bold***

5.4.3.5.3 XML Syntax

```
<!-- 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" use="required"/>
            <xs:attribute name="bandwidth" type="xs:unsignedInt" use="required"/>
            <xs:attribute name="trickModeApr" type="xs:double"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

5.4.3.6 Subsets

5.4.3.6.1 Overview

Subsets are described by the **`Subset`** element.

Subsets provide a mechanism to restrict the combination of *active* Groups where an active Group is one that for which the DASH client is presenting at least one of the contained Representations.

A Subset defines a set of one or more Groups. With the presence of a **`Subset`** element a **`Period`** element expresses the intention of the creator of the Media Presentation for a client to act as follows: At any time, the set of *active* Groups shall be a subset of one of the specified Subsets. Any Group not explicitly contained in any Subset element is implicitly contained in all specified Subsets.

This implies that

— Empty Subsets are not allowed.

— No Subset should contain all the Groups.

All Groups that are provided in the value of the `@group` attribute of a **`Contains`** element are contained in this Subset.

The semantics of the attributes and elements within a Subset are provided in Table 8 of 5.4.3.6.2. The XML-syntax of the Subset type is provided in 5.4.3.6.3.

5.4.3.6.2 Semantics

Table 8 — Subset Element Semantics

Element or Attribute Name	Use	Description
`Period`		
`Subset`	0...N	Provides Subset Information
`Contains`	1…N	Provide the information of contained Groups in this Subset.
`@group`	M	Specifies by its `@group` attribute, which Group is included in this Subset.
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 **@**.

NOTE: **It is an open question whether Subsets may also be used for selection of active Groups by the presentation layer. In this case an id and or annotation may be required to be added to the Subset element.**

5.4.3.6.3 XML Syntax

```
<!-- Subset selection information on Representation groups -->
<xs:complexType name="SubsetType">
    <xs:sequence>
        <xs:element name="Contains" type="ContainsType" minOccurs="1" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<xs:complexType name="ContainsType">
    <xs:attribute name="group" type="xs:unsignedInt" use="required"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

5.4.4 Segments and Segment Information

5.4.4.1 General

A Segment is defined as a unit that can be referenced by an HTTP-URL included in the MPD, where an HTTP-URL is defined as an <absolute-URI> according to RFC 3986, Subclause 4.3, with a fixed scheme of "http://" or https://, possibly restricted by a byte range if the **`Url`**`@range` attribute is provided. The byte range is expressed as a `byte-range-spec` as defined in RFC 2616, Subclause 14.35.1. It is restricted to a single expression identifying a contiguous range of bytes.

The Segments referenced through the HTTP-URLs in the MPD typically have assigned an *availability duration*, i.e. a time window in wall-clock time at which the Segments can be accessed at the resource specified by the HTTP-URL. The *availability duration* window is described by an *availability start time* and an *availability end time* for each resource.

Each **`Representation`** element shall contain at most one **`SegmentInfo`** element that together with the possibly present elements **`Period.SegmentInfoDefault`**, **`Group.SegmentInfoDefault`** and **`MPD.BaseURL`** as well as a document base URI shall contain sufficient information to determine the *Segment Information*.

If either of the elements **`Period.SegmentInfoDefault`** or **`Group.SegmentInfoDefault`** are present, then combination rules with the information in **`SegmentInfo`** apply to obtain a derived structure for the **`SegmentInfo`** element. These rules are provided in 5.4.4.2.

The rules to determine the *Segment Information* based on a **`SegmentInfo`** element are provided in 5.4.4.3. These rules apply to the derived **`SegmentInfo`** element after application of the rules in 5.4.4.2.

The semantics of the attributes and elements for the Segment and Segment Default Information are provided in 5.4.4.5, Table 10 and Table 11, respectively. The XML-syntax of the Segment Information is provided in 5.4.4.6.

5.4.4.2 Combination Rules to obtain Derived SegmentInfo Element

Reference resolutions as defined in 5.2.3 and base URL selection as defined in 5.2.4 shall be applied to obtain a base URL value.

The following rules apply for the combination of **Period.SegmentInfoDefault**, **Group.SegmentInfoDefault** and **Representation.SegmentInfo** to obtain a derived **SegmentInfo** element.

First, an attempt is made to produce a derived **SegmentInfoDefault** element:

— If a **Group.SegmentInfoDefault** is present then this element is the derived **SegmentInfoDefault** element.

— If a **Group.SegmentInfoDefault** is not present, but a **Period.SegmentInfoDefault** is present then this **Period.SegmentInfoDefault** is the derived **SegmentInfoDefault** element.

— If no derived **SegmentInfoDefault** element has been produced, there must be a **Representation.SegmentInfo** element present and it is used as the derived **SegmentInfo** element.

At this point if the derived **SegmentInfo** element has been produced, processing is complete. Otherwise, the following applies:

— If no derived **SegmentInfoDefault** element has been produced, the MPD is invalid and processing stops.

— If no **Representation.SegmentInfo** element is present, then the following applies for mapping the derived **SegmentInfoDefault** element to the derived **SegmentInfo** element:

  — Each element or attribute that is present in the derived **SegmentInfoDefault** is assigned to the corresponding element or attribute (with the same name) in the derived **SegmentInfo** element. This applies for **SegmentTimeline, InitialisationSegmentURL,** @duration, and @startIndex.

  — A possibly present **SegmentInfoDefault**@sourceURLTemplatePeriod attribute is assigned to the **SegmentInfo.URLTemplate**@sourceURL attribute.

  — A possibly present **SegmentInfoDefault**@indexTemplate attribute is assigned to the **SegmentInfo.URLTemplate**@indexURL attribute.

— If a **Representation.SegmentInfo** element is present then the following applies for combining it with the derived **SegmentInfoDefault** to produce the derived **SegmentInfo** element.

  — Each element or attribute that is present in the derived **SegmentInfoDefault** and not present in **Representation.SegmentInfo** is assigned to the corresponding element or attribute (with the same name) in the derived **SegmentInfo** element. This applies for **SegmentTimeline, InitialisationSegmentURL,** @duration, and @startIndex.

  — If **SegmentInfo.URLTemplate**@sourceURL is not present, but **SegmentInfoDefault**@sourceURLTemplatePeriod attribute is present, then this latter attribute us assigned to the attribute **SegmentInfo.URLTemplate**@sourceURL.

  — If **SegmentInfo.URLTemplate**@indexURL is not present, but **SegmentInfoDefault**@indexTemplate attribute is present, then this latter attribute us assigned to the attribute **SegmentInfo.URLTemplate**@indexURL.

  — If **SegmentInfoDefault** and **Representation.SegmentInfo** both contain one or more base URLs (specified using the **BaseURL** element), then this creates a level for reference resolution as defined in 5.2.3.

5.4.4.3 Segment Information based on Derived `SegmentInfo` element

5.4.4.3.1 Overview

The *Segment Information* is described in a **`SegmentInfo`** element (or in an derived structure after application of the rules in 5.4.4.2) and provides the following information:

- the presence or absence of Initialisation and Index Segment information
- the HTTP-URL and byte range for each accessible Segment in each Representation,
- for live services, the segment availability start time and segment availability end time of each Segment relative to the *PeriodStart* time,
- the approximated media start time of a Media Segment in the media presentation timeline within the Period,
- optionally, the byte ranges for Index information within a Media Segment

Any Segment URL, regardless whether provided explicitly or for example derived through Template-based Segment URL construction as defined in 5.4.4.4 may be relative or absolute. The resulting URLs shall be resolved with respect to a derived **`SegmentInfo.BaseURL`** element as defined in 5.2.3. Furthermore, if a derived **`SegmentInfo.BaseURL`** element is present, then alternative base URL selection as defined in 5.2.4 shall be applied.

The derivation of Initialisation, Media and Index Segment Information is provided in 5.4.4.3.2, 5.4.4.3.3, and 5.4.4.3.4, respectively.

5.4.4.3.2 Initialisation Segment Information

Each Representation has assigned at most one Initialisation Segment. The Initialisation Segment is conceptually processed by the media engine in Figure 2 to initialise the media engines for enabling play-out of Media Segments of the containing Representation.

The presence of an Initialisation Segment is indicated either

- by the presence of the **`InitialisationSegmentURL`** element that contains the URL to the Initialisation Segment, possibly restricted by a byte range,
- or, by the presence of **`UrlTemplate`**`@sourceURL`. In this case the Template-based Segment URL construction in 5.4.4.4 shall be applied with Index set to 0 to obtain the URL to the Initialisation Segment.

If no Initialisation Segment URL is present for a Representation then each Media Segment within the Representation shall be self-initialising.

For live services, the segment availability start time of the Initialisation Segment is the *PeriodStart* time and the end time of the Initialisation Segment is the largest availability end time of any Media Segment in this Representation.

5.4.4.3.3 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:

- Media Segment URL, possibly restricted by a byte range
- index of the Media Segment in the Representation

— approximate start time of the Media Segment in the Representation

— approximate duration of the Media Segment

The following information in the **SegmentInfo** element determines these parameters: **UrlTemplate**, **Url**, **SegmentList** and **SegmentTimeline** elements as well as @duration and @startIndex attribute. The example segment list generation process as specified in B.3.3 generates a list of Media Segments based on this information.

The list of Media Segments shall contain at least one entry. Therefore, the derived **SegmentInfo** element shall contain exactly one of the following choices to determine the Media Segment URL and the Index of the Media Segment:

— one **UrlTemplate** element: In this case the Template-based Segment URL construction in 5.4.4.4 shall be applied with the Index of the Media Segment in the media segment list. The first index in the list is determined by the value of the **SegmentInfo**@startIndex attribute, if present, or is 1 in case this attribute is not present. The last index in the list is determined by the value of the **UrlTemplate**@endIndex attribute, if present, or is infinite in case this attribute is not present.

— one or more **Url** elements: In this case the **Url** is directly assigned to the Media Segment URL. The first index in the list is determined by the value of the **SegmentInfo**@startIndex attribute, if present, or is 1 in case this attribute is not present. The last index is the sum of the first index and the number of list entries.

— one or more **SegmentList** elements that itself contains a list of **Url** elements for a consecutive list of Media Segment URLs. The first index in the list is determined by the value of the **SegmentList**@startIndex attribute, if present, or is identical to **SegmentInfo**@startIndex in case this attribute is not present. The sequence of multiple **SegmentList** elements within a Representation shall result in Media Segment List with consecutive indices.

— none of the above: In this case the same procedure shall be followed as for a single **Url** element containing an empty @sourceURL attribute.

For the derivation of the approximate start time and duration of each Media Segment in the list of media segments, the index of the media segment the following information is used.

— If neither @duration attribute nor **SegmentTimeline** element is present, then the Representation shall contain exactly one Media Segment. The start time is 0 and the duration is obtained as it would be the last Media Segment in the Representation (see below for more details).

— If @duration attribute is present, then the approximate start time of the Media Segment is determined as (Index-1) times the value of the duration of the attribute @duration. The duration of the Media Segment is determined as the value of the attribute @duration except for the duration of the last Media Segment (see below for more details).

— If @duration attribute is not and the **SegmentTimeline** element is present then rules in 5.4.4.7 apply to determine the start time and duration of each Media Segment in the media segment list.

— To determine the duration of 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.

The start time is relative to the start of the Representation provided by the MPD. The start time is approximate and does not reflect the exact media time. However, the start time shall be drift-free between the time indicated in the MPD and internal clock of the Media Segments, i.e. the accuracy of the start time documented in the MPD relative to the internal clock does not depend on the position of the Segment in the Representation.

For live services, the segment availability start time of a Media Segment is the sum of the *PeriodStart* time and the start time of the Media Segment in the Representation. The availability end time of a Media Segment is the sum of the *PeriodStart* time, the start time of the Media Segment in the Representation, the durations of the Media Segment and the value of the attribute **MPD**@timeShiftBufferDepth.

5.4.4.3.4 Index Segment Information

Each Representation typically has assigned Index Information to describe subsegments and RAPs. This information is conceptually processed by the DASH access client in Figure 2 in order to access subsegments by the use of HTTP partial GET requests. Index Information may be available either in

— an explicitly declared Index Segment Information, or

— if no explicit Index Segment Information is available for a Representation, then each Media Segment itself within the Representation may contain indexing information.

The presence of explicit Index Segment information is indicated either

— by the presence of one or more **Index** elements. If more than one **Index** elements are provided then there must be the same number of **Index** elements as **Url** elements. In this case the *n*th **Index** provides the location of the index information for the *n*th Segment. If a single **Index** element is provided then this provides the location of the index information for all Segments. When no **Index**@sourceURL attribute is provided then the URL is implicitly specified by the **Url**@sourceURL attribute of the corresponding **Url** element (or the first **Url** element if only one **Index** element is provided).

— by the presence of **UrlTemplate**@indexURL attribute. In this case the Template-based Segment URL construction in 5.4.4.4 shall be applied with Index set to the Index of the corresponding Media Segment.

Index elements may also be used to provide the byte range for an index within a Media Segment, where this is allowed by the Media Segment format. In this case the @sourceUrl attribute shall not be present and the range specified shall lie completely within any byte range specified for the Media Segment.

The availability of Index Segments is identical to the availability to the Media Segments they correspond to.

5.4.4.4 Template-based Segment URL Construction

The derived **SegmentInfo** Element may contain **UrlTemplate** element. The **UrlTemplate**@sourceURL attribute and the **UrlTemplate**@indexURL attributes each represent a string that contains one or more of the identifiers as listed in Table 9. The attributes shall contain the *$Index$* identifier. If a **SegmentTimeline** element is present, the *$Time$* identifier may be substituted for the *$Index$* identifier.

A sub-string "$<Identifier>$" names a substitution placeholder matching a mapping key of "<Identifier>". In the request URL, the substitution placeholder shall be replaced by the substitution parameter as defined in Table 9. Substitution is performed left to right and identifier matching is case-sensitive. Unrecognized identifiers cause the URL formation to fail. 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.

It is the responsibility of the content author that the application if 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 1738.

Table 9 — Identifiers for URL Templates

$<Identifier>$	**Substitution parameter**
$$	Is an escape sequence, i.e. "$$" is replaced with a single "$"
$RepresentationID$	This identifier is substituted by the attribute **`Representation`**`@id` of the containing Representation.
$Index$	This identifier is substituted by the *Index* of the corresponding Segment. For an example DASH Client using this identifier to construct the list Media Segment URLs, refer to B.3.
$*Bandwidth*$	This identifier is substituted by the **`Representation`**`@bandwidth` attribute value.
$*Time*$	This identifier is substituted by the **`SegmentTimeline`**`@t` attribute value for the given Segment being accessed. Either $Index$ or $Time$ may be used but both identifiers shall not be used within the same URL Template.

5.4.4.5 Semantics Segment Information

Table 10 — Semantics of `SegmentInfoDefault` element

Element or Attribute Name	Use	Description
`SegmentInfoDefault`	0…1	provides default Segment information
`@duration`	O	Default duration of Media Segments
`@startIndex`	O	Default start index
`@sourceURLTemplatePeriod`	O	The string providing the media segment URL template on the level where the segment information is placed.
`@indexTemplate`	O	The string providing the index segment URL template on the level where the segment information is placed.
`InitialisationSegmentURL`	0…1	default for Initialisation Segment URL
`BaseURL`	0…N	Base URL element to be used for reference resolution
`SegmentTimeline`	0…1	Timeline of arbitrary media segment durations. See Table 10.
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 11 — Semantics of `SegmentInfo` element

Element or Attribute Name	***Use***	***Description***
`SegmentInfo`	1	Provides segment information.

@duration	O	If present, gives 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.
@startIndex	O	The index of the first accessible Media Segment in this Representation. In case of on-demand services or in case the first Media Segment of the Representation is accessible, then this value shall not be present or shall be set to 1.
BaseURL	0...N	Base URL element that can be used for reference resolution
SegmentTimeline	0...1	Timeline of arbitrary segment durations.
InitialisationSegmentURL	O	This element references the Initialisation Segment.
@sourceURL	O	The source string providing the URL. If not present, then any **BaseURL** element is mapped to the sourceURL attribute and the range attribute shall be present.
@range	O	The byte range restricting the above URL. If not present, the resources referenced in the sourceURL are unrestricted. The format of the string shall comply with the format as specified in 5.4.4.1.
UrlTemplate	0 … 1	The element includes attributes to generate a Segment list for the Representation associated with this element.
@sourceURL	O	The string providing the template to create the Media Segment List.
@indexURL	O	The string providing the template to create the Index segment List
@endIndex	O	The index of the last accessible Media Segment in this Representation. If not present the *endIndex* is unknown.
Index	0 ... N	Provides a set of explicit URL(s) of for Index Segment.
@sourceURL	O	The source string providing the URL. If not present, then the URL specified by the corresponding **Url** element is used.
@range	O	The byte range restricting the above URL. If not present, the resources referenced in the sourceURL are unrestricted. The format of the string shall comply with the format as specified in 5.4.4.1.

Url		0 ... N	Provides a set of explicit URL(s) for Segments. Note: The URL element may contain a byte range.
	@sourceURL	O	The source string providing the URL. If not present, then any **BaseURL** element is mapped to the @sourceURL attribute and the @range attribute shall be present.
	@range	O	The byte range restricting the above URL. If not present, the resources referenced in the sourceURL are unrestricted. The format of the string shall comply with the format as specified in 5.4.4.1.
SegmentList		0 … N	Provides a list of explicit URL(s) for Segments.
	@xlink:href	O	reference to external **SegmentList** element
	@xlink:actuate	OD default: "onRequest"	Processing set can be either "onLoad" or "onRequest"
	@startIndex	O	start index for this URL list.
	Url	0 … N	list of Media Segment URLs
	Index	0 … N	list of Index Segment URLs
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 @.			

5.4.4.6 XML Syntax

```
<!-- Default Segment access information -->
   <xs:complexType name="SegmentInfoDefaultType">
        <xs:sequence>
            <xs:element name="InitialisationSegmentURL" type="UrlType" minOccurs="0"/>
            <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="SegmentTimeline" type="SegmentTimelineType" minOccurs="0"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attributeGroup ref="SegmentInfoAttrGroup"/>
        <xs:attribute name="sourceURLTemplatePeriod" type="xs:string"/>
        <xs:attribute name="indexTemplate" type="xs:string"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
   </xs:complexType>

   <!-- Segment access information -->
   <xs:complexType name="SegmentInfoType">
        <xs:sequence>
            <xs:element name="InitialisationSegmentURL" type="UrlType" minOccurs="0"/>
            <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="SegmentTimeline" type="SegmentTimelineType" minOccurs="0"/>
            <xs:choice minOccurs="0">
                <xs:element name="UrlTemplate" type="UrlTemplateType" minOccurs="0"/>
```

```
                <xs:sequence>
                    <xs:element name="Url" type="UrlType" maxOccurs="unbounded"/>
                    <xs:element name="Index" type="UrlType" maxOccurs="unbounded"/>
                    <xs:any namespace="##other" processContents="lax" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:element name="SegmentList" type="SegmentListType" minOccurs="0"/>
                <xs:any namespace="##other" processContents="lax" minOccurs="0"
                     maxOccurs="unbounded"/>
            </xs:choice>
        </xs:sequence>
        <xs:attributeGroup ref="SegmentInfoAttrGroup"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <!-- grouping attributes common to SegmentInfo and SegmentInfoDefault -->
    <xs:attributeGroup name="SegmentInfoAttrGroup" >
        <xs:attribute name="duration" type="xs:duration"/>
        <xs:attribute name="startIndex" type="xs:unsignedInt" default="1"/>
    </xs:attributeGroup>

    <!-- A Segment URL  -->
    <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>

    <!-- A URL template  -->
    <xs:complexType name="UrlTemplateType">
        <xs:sequence>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="sourceURL" type="xs:anyURI"/>
        <xs:attribute name="indexURL" type="xs:anyURI"/>
        <xs:attribute name="endIndex" type="xs:unsignedInt"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <!-- SegmentList allows xlink in addition to list of URLs -->
    <xs:complexType name="SegmentListType">
        <xs:sequence>
            <xs:element name="Url" type="UrlType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Index" type="UrlType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute ref="xlink:href"/>
        <xs:attribute ref="xlink:actuate" default="onRequest"/>
        <xs:attribute name="startIndex" type="xs:unsignedInt"/>
    </xs:complexType>
```

5.4.4.7 Arbitrary Segment Duration Timelines

5.4.4.7.1 General

The **`SegmentTimeline`** element provides a way to express non-regular Representation segment durations while employing run-length compression for compactness when spans of segments have constant duration. The timeline allows the client to map segment indexes to times within the Period.

A single **`SegmentTimeline`** element may be contained in a derived **`SegmentInfo`** element.

5.4.4.7.2 Description

Times and durations in the timeline are expressed as rational numbers in seconds. The value of the `@timescale` attribute provides the denominator while the `@time` and `@duration` values provide the numerator.

The **`SegmentTimeline`** element contains a list of **`S`** elements with each specifying a required start time attribute `@t`, a required duration attribute `@d`, and an optional repeat count attribute `@r` for a number of times the duration is to be repeated for a contiguous series of Segments. The repeat value is zero based and defaults to zero (a single Segment in the series). For example, a repeat count of three means there are four Segments in the series.

The *S* elements are sorted by their `@t` attribute values. The time values are relative to the beginning of the Period. If the first `@t` value in the first **`S`** element is non-zero, the value given establishes the base time that is aligned with the start of the period.

When a **`URLtemplate`** is in force, the template always contains either a $Time$ or $Index$ identifier but not both. The $Time$ *identifier* is replaced by the segment start time (in `@timescale` units) obtained from the **`SegmentTimeline`**. If the $Index$ substitution tag is used, the substitution rules are the same as when a **`SegmentTimeline`** is not being used.

The semantics of the attributes and elements for Segment Timeline are provided in 5.4.4.7.3, Table 12. The XML-syntax of the Segment Timeline is provided in 5.4.4.7.4.

5.4.4.7.3 Schema Overview Table

Table 12 — Semantics of `SegmentTimeline` element

Element or Attribute Name	*Use*	*Description*
`SegmentTimeline`	0 ... 1	Segment timeline information
`@timescale`	M	The number of timeline units per second. This forms the denominator for expressing time in seconds as a rational number.
`S`	1 .. N	Time line Segment series element. Each series specifies a start time, duration, and number of times the duration is to be repeated for a contiguous series of Segments. These elements are sorted by their *t* attribute values.
`@t`	M	Time, in `@timescale` units, the first Segment in the series starts relative to the beginning of the period.
`@d`	M	Duration, in `@timescale` units, of each Segment in the series.
`@r`	OD default: 0	Repeat count of the number of Segments in the series after the first and with the same duration `@d`. This value is zero based so a value of three means four Segments in the series.

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 **@**.

5.4.4.7.4 XML Syntax

```
<!-- SegmentTimeline of arbitrary segment durations -->
<xs:complexType name="SegmentTimelineType">
    <xs:sequence>
        <xs:element name="S" minOccurs="1" maxOccurs="unbounded" >
            <xs:complexType>
                <xs:attribute name="t" type="xs:unsignedInt" use="required"/>
                <xs:attribute name="d" type="xs:unsignedInt" use="required"/>
                <xs:attribute name="r" type="xs:unsignedInt" use="optional" default="0"/>
                <xs:anyAttribute namespace="##other" processContents="lax"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="timescale" type="xs:unsignedInt" use="required"/>
</xs:complexType>
```

5.5 Program Information

5.5.1 Overview

Descriptive information on the program may be provided for each period within the **ProgramInformation** element. Attributes are provided to specify a URL for the mode information, the title, the source of the program, and some copyright information.

The semantics of the attributes within the **ProgramInformation** element are provided in Table 13 of 5.5.2. The XML-syntax of **ProgramInformation** element is provided in 5.5.3.

5.5.2 Semantics

Table 13 — Program Information Semantics

Element or Attribute Name	Use	Description
ProgramInformation	0...1	Provides descriptive information about the program
@moreInformationURL	O	If specified, this attribute contains an absolute URL which provides more information about the Media Presentation in this Period.
Title	O	May be used to provide a title for the Media Presentation
Source	O	May be used to provide information about the original source (for example content provider) of the Media Presentation.
Copyright	O	May be used to provide 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 @.

5.5.3 XML Syntax

```
<!-- 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="moreInformationURL" type="xs:anyURI"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
```

5.6 Additional Content Description Elements

5.6.1 General

Each Representation or per default each Group may have assigned certain content description elements. The content description elements are all structured in the same way, namely they contain a `@schemeIdUri` attribute to identify the scheme and optional **`SchemeInformation`** elements. The `@schemeIdUri` provides a URI to identify the scheme. The definition of this element is specific to the scheme employed. The scheme may be a URN or a URL. Specific elements for content description are defined for:

— **`ContentProtection`**: content protection scheme element.

— **`Accessibility`**: accessibility component scheme element.

— **`Rating`**: rating scheme element.

— **`Viewpoint`**: viewpoint scheme element.

— **`FramePacking`**: frame packing arrangement scheme element

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 content description elements with appropriate scheme information. However, in 5.6.4 we provide some specific scheme identification rules.

Content Descriptions on Representation Groups should be applied such that the information is sufficient to judge whether at least one Representation contained in this Representation Group can be accessed.

The semantics of the attributes within a Generic Content Descriptor element are provided in Table 14 of 5.6.2. The XML-syntax of a Generic Content Descriptor element is provided in 5.6.3. The specific descriptors follow these syntax and semantics.

5.6.2 Semantics of Generic Content Descriptor

Table 14 — Semantics of generic `ContentDescriptor` element

Element or Attribute Name	Use	Description
`ContentDescriptor`	O	This element provides information about the use of content description for this representation.
`@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

SchemeInformation	O	This element may provide the information about the used content description scheme and may be extended by the owners of the identified scheme to provide scheme specific 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 @.

5.6.3 XML-Syntax of Generic Content Descriptor

```
<!-- Generic named descriptive information about the content -->
<xs:complexType name="ContentDescriptorType">
    <xs:sequence>
        <xs:element minOccurs="0" name="SchemeInformation" type="xs:string"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="schemeIdUri" type="xs:anyURI" use="required"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

5.6.4 URIs to Identify Schemes

5.6.4.1 General

Generally, the declaration of URIs for specific schemes (both syntax and semantics) to be used in any of the Content Description Elements requires the definition of the URI by the authors that link the content description to the Media Presentation. However, in the following certain scheme identifiers are defined in this standard to enable usage of existing schemes in combination with this standard.

5.6.4.2 Content Protection

The following rules apply for the URIs to identify specific content protection schemes, i.e. schemes contained in the **ContentProtection** element:

— For Representations based on ISO/IEC14496-12, the following URIs are defined to indicate the Scheme Type specified in the Scheme Type Box within the Protection Scheme Information Box of the file:

`urn:mpeg:mpegB:dash:mp4protection:<Scheme Type>`

where `<Scheme Type>` is the 4CC contained in the Scheme Type Box, suitably escaped according to RFC 2141.

— For representations based on ISO/IEC 13818-1 (MPEG-2 Transport), the following URIs are defined to indicate the Conditional Access System used:

`urn:mpeg:mpegB:dash:mp2protection:<CA_system_ID>`

where `<CA_system_ID>` is the 4-digit lower-case hexadecimal representation of the 16-bit CA_system_ID from the `CA_descriptor` as defined in ISO/IEC 13818-1.

— The **SchemeInformation** element within the **ContentProtection** element, if present, may contain scheme specific information defined for the scheme identified by the URI in the `@schemeIdUri` attribute.

— A content protection scheme using the Protection System Specific Header Box defined in ISO/IEC14496-12 may be identified in the **ContentProtection** element by a UUID URN as defined in RFC 4122 indicating the UUID specified in the SystemId field of the Protection System Specific Header Box. This

does not imply that such schemes cannot define alternative URNs, or that all UUID URNs refer to schemes of this type.

5.6.4.3 Frame Packing

The following rules apply for the URIs to identify specific content protection schemes, i.e. schemes contained in the **`FramePacking`** element:

— For representations that contain a video component, the URI

`urn:mpeg:mpegB:framePacking:dash:14496:10:frame_packing_arrangement_type:<value>`

is defined to indicate the frame packing scheme as defined by Table D-8 of ISO/IEC 14496-10 ('Definition of frame_packing_arrangement_type'). The `<value>` shall be the 'Value' column as specified in Table D-8 and shall be interpreted according to the 'Interpretation' column in the same table.

— For representations that contain a video component, the URI

`urn:mpeg:mpegB:framePacking:dash:13818:2:stereo_video_format_type:<value>`

is defined to indicate the frame packing scheme as defined by Table L-1 of ISO/IEC 13818-2 ('Definition of stereo_video_format_type'). The `<value>` shall be the '`stereo_video_format_type`' column as specified in Table L-1 and shall be interpreted according to the 'Meaning' column in the same table.

5.7 Other Representation and Group Elements

5.7.1 Trick Mode

5.7.1.1 Overview

Representations may be assigned a **`TrickMode`** element. If present it indicates that this Representation may be advantageously used for trick modes, such as fast forward.

The semantics of the attributes within the **`TrickMode`** element are provided in Table 15 in 5.7.1.2. The XML-syntax of Trick Mode element is provided in 5.7.1.3.

5.7.1.2 Semantics

Table 15 — Semantics of `TrickMode` element

Element or Attribute Name	Use	Description
`TrickMode`	O	Provides the information for trick mode. It also indicates that the Representation may be used as a trick mode Representation.
`@alternatePlayoutRate`	OD default 1	Specifies the maximum playout rate as a multiple of the regular playout rate, which this Representation supports with the same decoder profile and level requirements as the normal playout rate.
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 @.		

5.7.1.3 Syntax: XML-Element

```
<!-- Gives information about trick mode -->
<xs:complexType name="TrickModeType">
    <xs:sequence>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="alternatePlayoutRate" type="xs:string"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

5.7.2 Multiple Views Descriptors

5.7.2.1 Overview

A presentation can contain two or more views of which one or more view pairs can contain stereoscopic content suitable to be displayed as left and right view. The **`MultipleViews`** element supports signalling of frame packing arrangement and the transport of multiple views in a single representation. It is also used to identify left or right view if a single view is contained per representation.

The semantics of the attributes within the **`MultipleViews`** element are provided in Table 16 of 5.7.2.2. The XML-syntax of **`MultipleViews`** element is provided in 5.7.2.3.

5.7.2.2 Semantics

Table 16 — Semantics of `MultipleViews` element

Element or Attribute Name	Use	Description
`MultipleViews`	0 ... 1	Provides information for 3D video in the representation
`@stereoId`	O	When present, indicates that the presentation contains a view that belongs to one or more stereo video pairs. If more than one stereo pair is available for a segment, each stereo pair is indicated by an index i. The `@stereoId` attribute string contains a whitespace separated list of substrings starting with either 'l' or 'r', followed by zero or more digits that form an optional index value i. If one substring equals 'ri' it indicates the right view, if 'li', it indicates the left view of the i-th stereo video pair. If only two views are available for a segment, the index value is an empty string.
`@maximumViews`	O	When present, indicate the maximum number of target output views of the operation points the representation can provide
`FramePacking`	0 .. N	A Content Descriptor, when present, indicates that the representation is a bitstream with frame packing arrangement information. For details see 5.6.
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 @.		

5.7.2.3 XML Syntax

```
<!-- Type for MultipleViews -->
```

```
<xs:complexType name="MultipleViewsType">
    <xs:element name="FramePacking" type="ContentDescriptorType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:attribute name="stereoId" type="StringVectorType"/>
    <!-- stereoid: list of strings starting with "l" or "r" followed by optional index number -->
    <xs:attribute name="maximumViews" type="xs:unsignedInt" default="1"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

5.8 Media Presentation Description Updates

For any time instant when the MPD document is available on the server, the server should maintain MPD validity in the sense that an example client using the segment list generation process as provided in Annex B.3 would be able to access any available Segment in the Segment list for any Representation.

The server may update the MPD document during the Media Presentation.

When the MPD is updated any Representation with the same **`Representation`**`@id` and within the same Period as a Representation appearing in the previous MPD shall be the identical to the previous one in the sense that all Representation attributes are identical and all segments with the same indexes within one Representation are identical.

Furthermore, if the MPD is updated, then the updates to the MPD shall be such that the updated MPD is compatible with the previous MPD in the following sense: An example client using the segment list generation process as provided in Annex B.3 would generate an identically functional Segment List from the updated MPD for any time up to the *CheckTime* as defined in Annex B.3.4 of the previous MPD as it would have done from the previous MPD.

The requirement ensures that

— clients may immediately begin using the new MPD without synchronisation with the old MPD, since it is compatible with the old MPD before the update time; and

— the update time does not need to be synchronised with the time at which the actual change to the MPD takes place: i.e. changes to the MPD may be advertised in advance.

6 Segment Formats

6.1 General

6.1.1 Segment Properties

The Segment formats specify the formats of the entity body of the request response when issuing a HTTP GET request or a partial HTTP GET with the indicated byte range through HTTP/1.1 as defined in RFC 2616 to a resource identified in the Media Presentation Description.

The Initialisation Segment contains initialisation information for accessing the Representation. The Initialisation Segment shall not contain any media data.

A Media Segment contains media components that are either described within this Media Segment or described by the Initialisation Segment of this Representation or both. In addition, a Media Segment

1) is assigned an MPD URL Element.

2) is explicitly or implicitly assigned a start time relative to the start of the Representation provided by the MPD. The client can therefore download the appropriate Segments in regular play-out mode or after seeking. The start time shall be drift-free between the time indicated in the MPD and internal

clock of the Media Segments, i.e. the accuracy of the start time documented in the MPD relative to the internal clock does not depend on the position of the Segment in the Representation.

3) should contain at least one representation access point (RAP). Note that certain profiles may mandate the presence of at least one RAP for each media segment.

4) should provide information how to access the Media Presentation within this Segment, e.g. exact timing, index. There is no requirement that a Media Segment starts with a RAP, but it is possible to signal in the MPD that all Segments within a Representation start with a RAP. The first Media Segment of a Representation shall always start with a RAP.

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 RAP. The time-accuracy enables a client to seamlessly switch Representations and jointly present multiple Representations.

6) may contain indexing information for randomly accessing subsegments of the Segment by using partial HTTP GET requests. Indexing information may also be provided in separate Index Segments. A generic syntax and semantic for indexing is for example provided by the Segment Index ('`sidx`') in ISO/IEC 14496-12.

7) may contain additional subsegment indexing information for accessing different levels of subsegments in a media segment. A generic syntax and semantic for sub-segment indexing is for example provided by the Subsegment Index ('`ssix`') in ISO/IEC 14496-12.

8) if first in a Representation, shall be assigned presentation time 0 in the Media Presentation timeline relative to the Period start time. If the first sample of any of the media components has a Media Presentation time relative to the Period start time greater than 0, then this presentation time shall be signalled in the Media Segment. No sample of any of the media components shall have Media Presentation time relative to the Period start time smaller than 0.

This standards focuses on segment formats based on MPEG container formats. Specifically

— in 6.2, segment formats are described when used with media segments based on the ISO Base Media File Format as defined in ISO/IEC 14496-12;

— In 6.3, segment formats are described when used with media segments based on the MPEG-2 Transport Stream as defined in the ISO/IEC 13818-2 Format;

In both case the segment formats are defined such that the media segment formats comply to the respective container formats.

6.1.2 Guidelines for adding delivery formats to DASH

In order to support use with DASH, a delivery format should have the property that decoding and playback of any portion of the media can be achieved using a subset of the media which is only a constant amount larger than the portion of the media to be played.

For example, a delivery format where the media is stored as a header followed by a sequence of small blocks, with the property that any block can be decoded and played out given only that block and the header has this property. The definition of these blocks and the mapping to the sub-segments in this standard is encouraged. A subsegment may be defined as a contiguous time interval of a segment and a contiguous byte range of a segment, for which no overlap in both dimensions with any other subsegment in the segment exists.

Additionally, it is desirable that the delivery format supports some kind of "index" which enables the byte range within the Segment corresponding to any given time range to be efficiently discovered. A suitable unit is the indexing of sub-segments. It should be possible to discover the position in the Segment of the index without downloading the whole Segment. The position of the index may also be advertised in the MPD or the index

may be provided as a separate Index Segment. The Segment Index ('sidx') defined in ISO/IEC 14496-12 may serve as a starting point.

6.2 Segment formats for ISO Base Media File Format

6.2.1 Introduction

This Clause introduces segment formats that are suitable to be used if Media Segments comply with the ISO Base Media File Format. All segment formats defined in this Clause contain one or more boxes in accordance with the boxed structure of the ISO base media file format [ISO/IEC 14496-12]. Segment formats that may be used in DASH are defined within this Clause.

Subsegments for media segments based on the ISO base media file format are defined as a self-contained set of one or more consecutive movie fragments; a self-contained set contains one or more movie fragment boxes with the corresponding media data ('mdat') box(es), and each movie fragment ('moof') box immediately precedes its corresponding media data box.

6.2.2 Initialisation Segment Format

The Initialisation Segment is conformant with the ISO base media file format and the Dynamic Adaptive Streaming over HTTP (DASH) brand defined here and shall carry "dash" as a compatible_brand.

The Initialisation Segment shall consist of the "ftyp" box, the "moov" box, and optionally the "pdin" box. The tracks in the "moov" box shall contain no samples (i.e. the entry_count in the "stts", "stsc", and "stco" boxes shall be set to 0) and is thus very small in size. This reduces the start-up time significantly as the Initialisation 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 Initialisation Segment provides the client with the metadata that describes the encoding of the media content, specifically of the Representation. The client uses the information in the "moov" box to identify the available media components and their characteristics.

The Initialisation Segment shall not contain any "moof" box.

6.2.3 Media Segment Types

6.2.3.1 General

All media segments shall conform to the general definitions in 6.2.3.2. Additional type-specific constraints are provided further below in this Clause.

Further rules on media segments in combination with certain MPD attributes are provided in 7.2.

Note that media segments may conform to multiple types. Conformance can be expressed by adding the brand (4cc) to the 'styp' box as a compatible brand and, if applicable, as the major brand.

Unless specified otherwise, the referred boxes in this Clause are specified in ISO/IEC 14496-12.

6.2.3.2 General Format Type

A Media Segment conforming to the Media Segment Format for DASH shall carry 'msdh' as a compatible_brand and is defined as follows:

— Each Media Segment may contain a 'styp' box.

— 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 for media data that does not use external data references and the flag ‘default-base-is-moof’ shall also be set; absolute byte-offsets shall not be used for this media data. 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 may also be present; 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 box shall be the entire segment.

6.2.3.3 Indexed Media Segment

A Media Segment conforming to the Indexed Media Segment Format shall carry ‘msix’ as a compatible_brand and is defined as follows

— Each Media Segment shall comply to the general type as defined in 6.2.3.4 and in addition in each self-contained movie fragment, the movie fragment (‘moof’) box is immediately followed by its corresponding media data (‘mdat’)

— Each Media Segment shall contain at least one ‘sidx’ box. The first ‘sidx’ box shall be placed before any ‘moof’ box and shall document subsegments that span the composition time of the entire segment.

6.2.3.4 Sub-Indexed Media Segment

A Media Segment conforming to the Sub-Indexed Media Segment Format shall carry ‘sims’ in the Segment Type box (‘styp’) as a compatible_brand and is defined as follows:

— It shall conform to the indexed media segment format as specified in 6.2.3.3.

— The Subsegment Index box (‘ssix’) shall be present and shall follow immediately after the ‘sidx’ box that documents the same subsegment. This immediately preceding ‘sidx’ shall only index subsegments.

6.2.4 Self-Initialising Media Segment Formats

6.2.4.1 General

A Self-Initialising Media Segment conforms to the concatenation of an Initialisation Segment and a Media Segment.

6.2.4.2 General Format Type

The Self-Initialising Media Segment is conformant with the ISO base media file format and the Dynamic Adaptive Streaming over HTTP (DASH) brand and shall carry “dash” as a compatible_brand.

6.2.4.3 Indexed Self-Initialising Media Segment

The Index Self-Initialising Media Segment conforms to the concatenation of an Initialisation Segment and an Indexed Media Segment and shall carry '`dash`' as a compatible_brand.

6.3 Segment formats for MPEG-2 Transport Streams

6.3.1 Introduction

This Clause introduces segment formats that are suitable to be used if Media Segments comply with MPEG-2 TS in accordance with [SO/IEC 13818-1]. Segment formats that may be used in DASH in this context are defined within this Clause.

In context of this Clause, a sample is an access unit encapsulated into one or more MPEG-2 TS packets, which have the same PID value. A subsegment in this context is defined as an Indexed set of access units consecutive in decode order.

A reference stream is defined in context of this Clause as a continuous group of samples of the primary media component.

6.3.2 Media Segment Types and Formats

6.3.2.1 General

Type-specific constraints are provided further below in this Clause.

Further rules on media segments in combination with certain MPD attributes are provided in Clause 7.

6.3.2.2 Basic Media Segment

A media segment shall be a valid MPEG-2 TS, conforming to [ISO/IEC 13818-1]. In addition, the following conditions shall be met:

— Media segment boundaries shall be placed at MPEG-2 TS Packet boundaries.

— Media segment shall contain one and only one program.

— Concatenation of consecutive segments of the same representation shall yield an MPEG-2 TS, conforming to ISO/IEC 13818-1. However, such a concatenation may have PCR and continuity counter discontinuities, and may exceed the maximum limits on PCR frequency.

6.3.3 Initialisation Segment Types and Formats

An initialisation segment shall be a valid MPEG-2 TS, conforming to [ISO/IEC 13818-1], and its concatenation with any media segment shall yield a valid MPEG-2 TS with possible discontinuities in PCR and/or continuity counter. Initialisation segment shall contain one and only one program.

The MIME type of an Initialisation Segment is “video/m2t”.

The Initialisation Segment should contain Program Specific Information (such as PAT, PMT). It may also contain codec-specific and conditional access initialization information.

Initialisation segment is optional. If it is not present in a given Representation, all media segments belonging to this Representation shall be self-initialising. In this case, all initialisation information shall be present in the beginning of the segment, prior to any media data. In this case, there shall be no packets of any PID carrying any media, data, nor PCR's prior to the last packet of PSI. PSI shall have at least a PAT followed by a PMT.

Should both an initialisation segment be present, and initialisation data appear inside a media segment, initialisation information conveyed within the media segment overrides the respective initialisation information in the initialisation segment. E.g., information contained in a PAT within the media segment takes precedence over information in PAT within the initialisation segment.

No media segment shall depend on initialisation information in appearing in any preceding media segment.

6.3.4 Index Segment

6.3.4.1 Index Segment for a single Media Segment

An Index Segment that indexes exactly one Media Segment shall carry '`isss`' as a compatible_brand and is defined as follows:

— Each Index Segment shall begin with an '`styp`' box, and the brand '`isss`' shall be present in the '`styp`' box;

— Each Index Segment shall contain the Segment Index box(es) that indexes exactly one Media Segment.

6.3.4.2 Index Segment for multiple Media Segments

An Index Segment that indexes multiple Media Segments is defined as follows:

— Each Index Segment shall begin with an '`styp`' box, and the brand 'isms' shall be present in the '`styp`' box;

— Each media segment is indexed by one or more Segment Index Box(es); the boxes for a given media segment are contiguous;

— The indexing information for the media segments is concatenated in order, preceded by a single overall segment index box;

— The overall segment index box has one entry in its loop for each media segment, and each entry refers to the segment index information for a single media segment.

7 Media Presentation Authoring Rules

7.1 General

A Media Presentation is composed by an MPD and the referenced segments. Sections 5 and 6 define the formats for these two key components of a DASH-compatible Media Presentation. In this Clause, media presentation authoring rules are provided on how the MPD and different segment formats may be combined to establish a complete Media Presentation.

Specifically aspects are addressed that deal with the segment formats of Representations, that have special alignment with other Representations to simplify seamless switching and joint presentation.

A specification for how to use a media container format with DASH shall include:

— Definition of the MIME type for Segments based on the container format, together with MIME parameters to specify the codecs and codec parameters of the contained media (e.g. the codecs parameter of RFC 4281).

— Description of either a self-initialising Media Segment or the combination of an Initialisation Segment and a Media Segment format.

In addition, the specification may further define:

— Index Segments

— Interpretation of Representation Access Point (RAP), defined in 3, in the context of the media container format

— Container-format-specific semantics for the `@bitstreamSwitchingFlag`, `@segmentAlignmentFlag` and `@subsegmentAlignmentFlag`. These must align with the definitions in 5.4.2, 5.4.2, and 5.4.3.3, respectively.

Note that Representation attributes present in the MPD may also be repeated in the media itself, e.g. in an Initialisation Segment or a Media Segment. The media content shall be provided such that no mismatch between these two values occurs. If it does, the value in the media itself shall take precedence over values expressed in the MPD, especially when used in the media decoding process.

7.2 Media Presentation based on the ISO base media file format

7.2.1 Introduction

The Media Presentation as introduced in Clauses 5 and 6 is instantiated in this Clause using the ISO base media file format [ISO/IEC 14496-12]. This instantiation is referred to as “ISO BMFF-based DASH”. An ISO BMFF-based DASH media presentation is described by an MPD as specified in 5.1. The MIME type of the MPD shall be as defined in Annex D.

The `@mimeType` attribute of each Representation (either contained in the **`Representation`** or **`Group`** element) shall be provided according to RFC 4281.

The following segment types and formats may be used

— Initialisation Segments complying with formats as defined in 6.2.2.

— Media Segments complying with formats as defined in 6.2.3.2.

— Self-Initialising Media Segments complying with formats as defined in 6.2.4.

For ISO BMFF-based DASH the following applies:

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

   i) The Initialisation Segment as defined in 6.2.2 shall be present.

   ii) Media Segments shall not be self-initialising. The Media Segment format is defined in 6.2.3.

2) In case a Representation contains only a single Media Segment, then either one of the following two options are valid.

   — One Initialisation Segment as defined in 6.2.2 and one Media Segment as defined in Clause 6.2.3.

   — One Self-Initialising Media Segment as defined in 6.2.4.3.

Index Segments shall not be present. However, an **`Index`** element may be present to signal the byte range for an index within a Media Segment as defined in 5.4.4.3.4.

The Representation Access Point follows the definition in 3.1, In addition, the latest index for a Representation Access Points is the first byte of a 'moof' box of a movie fragment that contains a Random Access Point and the presentation time is the composition time of the first Random Access Point in this movie fragment.

The content authoring rules for the media segments when setting certain flags are set to true or false for ISO BMFF-based DASH are provided in 7.2.2.

In case Sub-Representations are used, the rules in 7.2.3 shall apply.

7.2.2 Authoring Rules for specific MPD flags

7.2.2.1 Segment Alignment

If the `@segmentAlignmentFlag` is set to '`true`', the requirements stated in 5.4.2 conditions shall be met.

7.2.2.2 Bitstream Switching

If the `@bitstreamSwitchingFlag` is set to '`true`', for a set of Representations (in the **`Period`** element or **`Group`** element), the conditions stated in 5.4.2 and all following conditions shall be satisfied:

— The `@segmentAlignmentFlag` for the same set of Representations shall be set to '`true`' and the value of the `@duration` attribute on Representation level shall be identical for all Representations in the Period.

— The `@startWithRAP` flag shall be set to `'true'`

— Let X be the concatenation of an Initialisation Segment with consecutive Media Segments of a single Representation within a Period, starting with the first Media Segment, and Y be the concatenation wherein the Media Segments with the same constraints come from at least two Representations within the same group. The following shall apply to all possible instances of X and Y: for any media sample commonly present in X and Y, the decoding time and presentation time derived when playing X shall be identical to the decoding time and presentation time, respectively, derived when playing Y, by a legacy player.

7.2.2.3 Subsegment Alignment

If the `@subsegmentAlignment` attribute is set to '`true`', the semantics as defined in 5.4.3.3 shall apply. In particular, for formats based on the ISO base media file format the following applies:

— streams are tracks as defined in the file format;

— presentation times are composition times.

7.2.3 Sub-Representations

If a **`SubRepresentation`** element is present in a Representation in the MPD, then the media segments in this Representation shall conform to a Sub-Indexed Media Segment as defined in 6.2.3.4. The Initialisation Segment shall contain the Level Assignment ('`leva`') box.

The attribute `@level` specifies the level to which the described Sub-Representation is associated to 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.

Note that the brand '`sims`' can for example be signaled in the **`Group`**`@mimeType`, **`Representation`**`@mimeType` or **`SubRepresentation`**`@mimeType` attribute.

7.3 Media Presentation based on MPEG-2 TS

7.3.1 Introduction

In this Clause, a Media Presentation is instantiated based on Media Segment Formats using the MPEG-2 TS as defined in ISO/IEC 13818-1. This instantiation is referred to as “MPEG-2 TS-based DASH”. A MPEG-2 TS-based DASH media presentation is described by an MPD as specified in Clause 5. The MIME type of the MPD shall be as defined in Annex D.

The `@mimeType` attribute of each media segment Representation is “`video/MP2T`”.

The following segment types and formats may be used

— Initialisation Segments complying with formats as defined in 6.3.3.

— Media Segments complying to formats as defined in 6.3.2.

— Index Segments complying to formats as defined in 6.3.4.

  — If a single Index Segment indexes a single Media Segment, Index Segments complying with format as defined in 6.3.4.1 shall be used.

  — If a single Index Segment indexes all the Media Segments in a representation, Index Segments complying with the format as defined in 6.3.4.2 shall be used.

Sub-Representations are not defined for MPEG-2 TS-based DASH. Therefore, the **`SubRepresentation`** element shall not be present if the Representation uses MPEG-2 TS as Media Segment format.

7.3.2 Authoring Rules for specific MPD flags

7.3.2.1 Starting from a Random Access Point

If `@startWithRAP` flag is set to '`true`' for a Representation or a group of Representations, then *all* media segments of each of this Representation shall start with a RAP. In this case, all of the following conditions shall be met:

— The first packet of the PCR PID shall precede the first packet carrying any media data. If PCR’s are carried on a media PID, the first packet of this PID shall be the first packet following the initialisation data, and shall carry a PCR.

  Note: In order to avoid changing the underlying MPEG-2 TS, the implementer may choose to add a packet carrying only adaptation field with a PCR, but no payload.

— The first access unit of the primary media component shall be an Elementary Stream Random Access Point as defined in [ISO/IEC 13818-1].

— The payload of the first PES packet shall contain the access unit above, and it shall contain a PTS timestamp.

7.3.3 Bitstream Switching

If `@bitstreamSwitchingFlag` is set to '`true`', the following conditions shall be met:

— The `@segmentAlignmentFlag` for the Period shall be set to ‘`true`’ and the value of the `@duration` attribute on Representation level shall be identical for all Representations in the Period.

— A concatenation of any two consecutive media segments from *different* Representations, which are prepended by their respective Initialisation Segments (if present), shall yield a valid MPEG-2 TS.

— Conditions in 7.3.2.1 shall be met.

7.3.4 Segment Alignment

If the `@segmentAlignmentFlag` is set to '`true`', the following conditions shall be met:

— The position of all segment boundaries within a Period is aligned in time across all representations.

— Media segment shall contain only complete PES packets, and the first PES packet shall contain a PTS timestamp. Media segment boundary shall correspond to a boundary between access units in the primary media component.

7.3.5 Subsegment Alignment

If the `@subsegmentAlignment` attribute is set to '`true`', the semantics as defined in 5.4.3.3 shall apply. In particular, for MPEG-2 TS-based DASH, the following applies:

— streams and samples are as defined in 6.3.1

— presentation times are those indicated in the index.

— all media segments shall be indexed;

David Singer 2/6/11 11:30 AM
Comment: we need to say something about alignment

7.3.6 Content Protection

All information necessary for decrypting the first encrypted TS packet in a segment shall be present before this packet, either in the same segment, and/or in the initialisation segment (if used). Therefore, CAT shall be present in PSI. The ECM necessary for descrambling the first scrambled packet of the segment shall be present within the segment before such a packet.

8 Profiles

8.1 Definition

Profiles of DASH are defined so as to enable interoperability and the signaling of the use of features etc.

A profile has an identifier and refers to a set of specific restrictions. Those restrictions might be on features of the media presentation description (MPD) document, usage of the network, media format(s), codec(s) used, protection formats, or on quantitative measures such as bit-rates, segment lengths, screen size, and so on. Profiles defined in this part of ISO/IEC 23001 define restrictions on features of this part of ISO/IEC 23001 only (e.g. not codec types). Externally defined profiles may additionally impose restrictions on other aspects of media delivery.

A profile is a claim and a permission; it claims that the media presentation (MPD document and segment formats) conforms to the profile, and gives permission to a reader that implements that profile to read the media presentation, interpret what it recognizes, and ignore the material it does not understand.

The profiles with which an MPD complies are indicated in the **`MPD`**`@profiles` attribute. This element is a space-delimited list of profile identifiers each of which is a URI. Profile identifiers defined in this specification are URNs conforming to RFC 3406. 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.

8.2 Full 2011 profile

The full profile is identified by the URN "urn:mpeg:mpegB:profile:dash:full:2011". The full profile includes all features and Segment Types defined in this specification.

8.3 ISO Base media file format basic on-demand profile

8.3.1 General

The basic on-demand profile is identified by the URN "urn:mpeg:mpegB:profile:dash:isoff-basic-on-demand" and makes use of the following features:

— Single period per media presentation

— ISO Base Media File Format segments

— Self-initializing indexed media segments (i.e. one segment per representation).

— Subsegment alignment is required

— Un-multiplexed streams

— Url diversity using the **BaseURL** element

— Content Protection and accessibility annotations

8.3.2 Media Presentation Description

Table 17 lists the MPD elements and attributes that must be supported for compliance to this profile:

Table 17 — Supported Media Presentation Description elements and attributes

Element or Attribute Name	Special requirements (if any)
MPD	
@type	Shall be "OnDemand"
@mediaPresentationDuration	Shall be present
@minBufferTime	Shall be present
BaseURL	
ProgramInformation	
Period	At most one shall be present
Group	One or more shall be present
ContentProtection	
@schemeIdUri	
SchemeInformation	
Accessibility	
@schemeIdUri	
SchemeInformation	
Viewpoint	
@schemeIdUri	
SchemeInformation	
Rating	
@schemeIdUri	
SchemeInformation	
@width	

@height	
@parx	
@pary	
@lang	
@mimeType	Shall be present and shall indicate a single media type
@frameRate	
@numberOfChannels	
@samplingRate	
@subsegmentAlignmentFlag	Must be '**true**'
Representation	
ContentProtection	
@schemeIdUri	
SchemeInformation	
@width	
@height	
@parx	
@pary	
@lang	
@frameRate	
@id	Shall be present
@bandwidth	Shall be present
@dependencyId	
@numberOfChannels	
@samplingRate	
TrickModeType	
@alternatePlayoutRate	
SegmentInfo	Shall be present
BaseURL	

8.3.3 Segment formats

Segments shall comply with Indexed self-initialising Media Segment as defined in 6.2.4.4 of this standard. All Segment Index boxes shall be placed before any Movie Fragment boxes. Where nested Segment Index boxes are used there shall be at most 3 levels (where a single Segment Index box referring directly to Movie Fragments is considered a single level).

Annex A
(informative)

Patent Statements

The International Organization for Standardization and the International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this part of ISO/IEC 23001 may involve the use of patents.

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured the ISO and IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patents right are registered with ISO and IEC. Information may be obtained from the companies listed below.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 23001 may be the subject of patent rights other than those identified in this annex. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

	Company
1.	

Annex B
(informative)

Example DASH Client Behaviour

B.1 Introduction

The information on client behaviour is purely informative and does not imply any normative procedures on client implementations. However, this information may serve as a guideline to better understand certain features of the formats in the normative parts of this specification.

B.2 Overview

A DASH client is guided by the information provided in the MPD. The description in this Annex assumes that the client has access to the MPD that it fetched at time *FetchTime*, the time at the client when the server executes the request for the fetch. The client typically should not used the time when it actually successfully received the MPD, but should take into account delay due to MPD delivery and processing. . The fetch is considered successful fetching either if the client either 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 provides a continuous streaming experience to the user:

1) The client parses the MPD and creates a list of accessible Segments for each Representation for the actual client-local time *NOW* taking into account the procedures specified in B.3.

2) The client selects one or multiple Representations based on the information in the Representation attributes and other information, e.g. available bandwidth and client capabilities. Representations assigned to group 0 are presented without any other Representation. Representations assigned to a non-zero group are typically presented in combination with Representations from groups other than their own, not including the group 0. The selected Representations may be constrained by the **`Subset`** element. For each selected Representation, the client acquires the Initialisation Segment, if present, and the Media Segments.

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

4) The client buffers media of for at least value of `@minBufferTime` attribute duration before starting the presentation, provide the observed throughput remains at or above the sum of the `@bandwidth` attributes of the selected Representations (if not, longer buffering may be needed).

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 throughput. With any request for a Media Segment containing a random access point, the client may switch to a different Representation.

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

i) The `@mediaPresentationDuration` attribute is not declared, or if any media described in the MPD does not reach to the end of the Media Presentation and

ii) the current playback time gets within a threshold (typically described by 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 *FetchTime*. Once received the client takes into account the possibly updated MPD and the new *FetchTime* in the regeneration of the accessible Segment list for each Representation.

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

B.3 Segment List Generation

B.3.1 General

Assume that the HTTP-streaming client has access to an MPD. This Clause describes how a client may generate a Segment list as shown in Table B.1 from an MPD 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 HTTP Streaming server, which is in turn synchronised to UTC, may experience issues in accessing segments due to availability. HTTP Streaming clients should synchronize their clocks to a globally accurate time standard.

Table B.1 — Segment List

Parameter Name	Cardinality	Description
Segments	1	Provides the Segment URL list.
InitialisationSegment	0, 1	Describes the Initialisation Segment. If not present each Media Segment is self-initialising.
URL	1	The URL where to access the Initialisation 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 approximate start time of the Media Segment in the Period relative to the start time of Period. To obtain the start time of the Media Segment in the Media Presentation, the start time of the Period needs to be added for On-Demand services. For live services, in addition also the value of the `@availabilityStartTime` attribute needs to be added.
URL	1	The URL where to access the Media Segment (the client would restrict the URL with a byte range if one is provided in the MPD).

Each **`SegmentInfo`** element together with possibly present **`Period.SegmentInfoDefault`** and/or **`Group.SegmentInfoDefault`** is used to generate a list of accessible Segments for each Representation as defined in 5.4.4. Specifically.

a) The client uses URI reference resolution as defined in 5.4.4 and 5.2.3. If the MPD is known to be supplied using a URL and over a suitable protocol, that URL establishes a base URL for the segments URLs within the MPD. There may be **`BaseURL`** elements on different levels. If the **`BaseURL`** elements supplied at any level are absolute, they give the base URLs for the levels below. Otherwise the base URLs for levels below are formed from the base URLs of the higher level composed with the value of each **`BaseURL`** element. Normal URL composition may be used, using relative URLs, which are composed against a base URL. The composition of a relative URL with an effective base URL is done using normal URL Reference Resolution (see RFC 3986, 5.2).

b) If the derived **SegmentInfo** element contains a template element is implied, then the procedures in B.3.2 are used to generate a list of Media Segments.

c) If the *SegmentInfo* element contains one or more **Url** elements (or one or more **SegmentList** elements containing **Url** elements) providing a set of explicit URL(s) for Media Segments, then the procedures in B.3.3 are used to generate a list of Media Segments.

d) If the **MPD**@type attribute is Live, then the restrictions on Media Segment Lists as provided in B.3.4 need to be taken into account.

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

B.3.2 Template-based Generation of Media Segment List

If the derived **SegmentInfo** element contains a template, then the procedures in this Clause are used to generate a list of Media Segment parameters, i.e. segment URLs and start times, and no byte ranges are associated with the URLs.

The Segment information for a Representation is obtained as defined in 5.4.4 and is collected in a derived **SegmentInfo** element.

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

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

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

If the derived **SegmentInfo** Element contains a **UrlTemplate** element, the relevant identifiers are replaced.

Assume now that Media Segments within a Representation have been assigned consecutive indices *i*=1,2,3…., i.e. the first Media Segment has been assigned the index *i*=1, the second Media Segment has been assigned the index *i*=2, and so on.

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

1) Set i=@startIndex.

2) The start time of the first Media Segment is obtained as (@startIndex-1)*@duration, i.e. MediaSegment.StartTime[1] = (@startIndex-1)*@duration. If the *duration* attribute is not provided, then the MediaSegment.StartTime[1] of the only provided Segment is set to 0.

3) The URL of the Media Segment *i*, MediaSegment.URL[*i*], is obtained by replacing the $*Index*$ identifier by *Index*[*i*] in the template. Furthermore, any relative URLs are resolved as specified in 5.4.4.2.

4) If ((*PeriodStart* + MediaSegment.StartTime[*i*] + @duration) <= *PeriodEndTime*) and if the attribute @endIndex is given and (*i* < @endIndex) then increment *i*, set MediaSegment.StartTime[*i*] = MediaSegment.StartTime[*i*-1] + @duration, and proceed with step 3. Otherwise, continue with step 5.

5) The restrictions as specified in B.3.4 are applied for the creation of the accessible list of Media Segments and this concludes Segment List generation.

If a **`SegmentTimeline`** element has been given, instead of the steps above the following steps are used:

1. A list of Segment start times and durations is first generated by expanding the *SegmentTimeline* list from its run-length compressed form into a *SegmentTimelineList* of StartTime and Duration values for each Segment. This new list is indexed starting at `@startIndex`.

2. Set *i* = `@startIndex`

3. MediaSegment.StartTime[*i*] = SegmentTimelineList[*i*].StartTime

4. MediaSegment.URL[*i*] is obtained by replacing any *$Index$* identifier with *i* or any *$Time$* identifier with MediaSegment.StartTime[*i*] in the template.

5. If ((*PeriodStart* + MediaSegment.StartTime[*i*] + SegmentTimelineList[*i*].Duration <= *PeriodEndTime*) and, if the attribute `@endIndex` is given and (*i* < `@endIndex`), then increment *i* and proceed with step 3. Otherwise, continue with step 6.

6. The restrictions as specified in B.3.4 are applied for the creation of the accessible list of Media Segments.This concludes the Segment list generation.

B.3.3 Playlist-based Generation of Media Segment List

If the derived **`SegmentInfo`** element contains one or more **`Url`** elements, then the procedures specified in this Clause apply to generate a valid list of accessible Media Segment URLs and start times.

Assume that Media Segments within a Representation have been assigned consecutive indices *i*=1,2,3…., i.e. the first Media Segment has been assigned *i*=1, the second Media Segment has been assigned *i*=2, and so on.

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

A valid list of Media Segments with segment indices *i*=`@startIndex`, `@startIndex`+1, …, MediaSegment.StartTime[*i*] and MediaSegment.URL[*i*] is obtained as follows:

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

2) The URL of the Media Segment *i*, *MediaSegment.URL[i]*, is obtained as the sourceURL attribute of the (*i*-`@startIndex` +1)th **`Url`** element in the **`SegmentInfo`** element taking into account URI reference resolution, restricted to the byte range specified in the range attribute of the same **`Url`** element, if present.

3) If the duration attribute is provided, then the *MediaSegment.StartTime[i]* of Media Segment i is obtained as (*i*-1)*`@duration`. If the duration attribute is not provided and a **`SegmentTimeline`** element is in effect then the *MediaSegment.StartTime[i]* is obtained as *SegmentTimelineList[i].StartTime*. Otherwise, the *MediaSegment.StartTime[1]* of the only provided Segment is set to 0.

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

5) The restrictions as specified in B.3.4 are applied for the creation of the accessible list of Media Segments. This concludes Segment list generation.

B.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*.

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

In addition, for live services, assume the variable *CheckTime* associated to an MPD with *FetchTime* is defined as:

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

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

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

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

B.4 Seeking

Assume that a client attempts to seek to a specific presentation time *tp* in a Representation with start time *PeriodStart*. *PeriodStart* defines the absolute start time of the Media Segment in the Media Presentation, i.e. for On-Demand services the start time of the Period needs to be added and for live services, in addition also the value of the *availabilityStartTime* attribute needs to be added. Before accessing the Media Segments of a Representation, the client needs to download the Initialisation Segment, if present.

Based on the MPD, the client has access to the Media Segment start time and Media Segment URL of each Segment in the Representation. The Segment index *segment_index* of the Segment most likely to contain media samples for presentation time *tp* is obtained as the maximum Segment index *i*, for which the start time MediaSegment[*i*].StartTime is smaller or equal to the presentation time relative to the Representation start time *tp* minus *PeriodStart*, The Segment URL is obtained as MediaSegment[*segment_index*].URL.

Note that timing information in the MPD may be approximate due to issues related to placement of Representation 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 *tp* 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 *tp* and the start of the retrieved Segment is always available.

For accurate seeking to a presentation time *tp*, the HTTP-Streaming Client needs to access a Represention Access Point (RAP). To determine the RAP in a Media Segment in case of DASH, the client may, for example, use the information in the Segment Index if present to locate the random access points and the corresponding presentation time in the Media Presentation. In the case that a Segment is a movie fragment, it is also possible for the client to use information within the 'moof' and 'mdat' boxes, for example, to locate Random Access Points in the media and obtain the necessary presentation time from the information in the movie fragment and the segment start time derived from the MPD. If no RAP with presentation time before the

requested presentation time *tp* is available, the client may either access the previous Segment or may just use the first RAP as the seek result. When Media Segments start with a RAP, these procedures are simple.

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

B.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 the MPD for a specific Representation contains the **`TrickMode`** element, then this Representation is explicitly enabled for the use with trick modes. The client may play the Representation with any speed up to the regular speed times the specified `@alternatePlayoutRate` attribute with the same decoder profile and level requirements as the normal playout rate.

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

The client may also use Sub-Representations for trick modes such as fast forward.

B.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 RAP in the “new” Representation at a desired presentation time *tp* later than and close to the current presentation time. Presenting the “old” Representation up to the RAP in the “new” Representation enables seamless switching.

If `@segmentAlignmentFlag` is set true, the client may switch at any segment boundary by just concatenating segments with consecutive indices from different Representations.

If `@subSegmentAlignmentFlag` is set true, the client may switch at any subsegment boundary by just concatenating subsegments at appropriate times from different Representations.

B.7 Reaction to Error Codes

The DASH access client provides a streaming service to the user by issuing HTTP requests for Segments at appropriate times. The DASH access 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 DASH access client receives an HTTP client or server error (i.e. messages with 4xx or 5xx error code), the client should respond appropriately (e.g. as indicated in RFC 2616) to the error code. In particular, clients should handle redirections (such as 301 and 307) as these may be used as part of normal operation.

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

If the DASH access client receives an HTTP client error (i.e. messages with 4xx error code) for the request of an Initialisation Segment, the Period containing the Initialisation Segment may not be available anymore or may not be available yet.

Similarly, if the DASH access 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 both these case the client should check if the precision of the time synchronization to a globally accurate time standard is sufficiently accurate. If the clock is believed accurate, or the error re-occurs after any correction, 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. If multiple **`BaseURL`** elements are available, the client may also check for alternative instances of the same content that are hosted on a different server.

B.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.

During long live streaming sessions, 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 follows. The pace r_1 of the encoder clock in relation to the UTC is recovered from Producer Reference Time boxes. If the relative pace r_1 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 r_2 of the receiver playout clock in relation to UTC is created by accessing a UTC source. A timescale multiplication factor c is equal to r_1/r_2. 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 C
(normative)

MPD Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"
    attributeFormDefault="unqualified"
    elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011">

    <xs:import namespace="http://www.w3.org/1999/xlink"
                          schemaLocation="http://www.w3.org/1999/xlink.xsd"/>

    <xs:annotation>
        <xs:appinfo>Media Presentation Description</xs:appinfo>
        <xs:documentation xml:lang="en">
            This Schema defines Media Presentation Description for MPEG DASH.
            The namespace identifier is for this draft version of the specification.
            The final version will use a different namespace identifier to align with 3GPP.
            However, changes from the draft to the final version are not expected to change
            the information model, even if the schema changes, and trial implementation and
            comment on the schema presented here are both encouraged.
        </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"/>
            <xs:element name="Period" type="PeriodType" maxOccurs="unbounded"/>
            <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="profiles" type="URIVectorType"/>
        <xs:attribute name="type" type="PresentationType" default="OnDemand"/>
        <xs:attribute name="availabilityStartTime" type="xs:dateTime"/>
        <xs:attribute name="availabilityEndTime" type="xs:dateTime"/>
        <xs:attribute name="mediaPresentationDuration" type="xs:duration"/>
        <xs:attribute name="minimumUpdatePeriodMPD" type="xs:duration"/>
        <xs:attribute name="minBufferTime" type="xs:duration"/>
        <xs:attribute name="timeShiftBufferDepth" 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="OnDemand"/>
            <xs:enumeration value="Live"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- 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="moreInformationURL" type="xs:anyURI"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
```

```
    </xs:complexType>

    <!-- Period of a presentation -->
    <xs:complexType name="PeriodType">
        <xs:sequence>
            <xs:element name="SegmentInfoDefault" type="SegmentInfoDefaultType" minOccurs="0"/>
            <xs:element name="Representation" type="RepresentationType"
                                    minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Group" type="GroupType"
                                    minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Subset" type="SubsetType" 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="start" type="xs:duration"/>
        <xs:attribute name="id" type="xs:string" />
        <xs:attribute name="duration" type="xs:duration"/>
        <xs:attribute name="minBufferTime" type="xs:duration"/>
        <xs:attribute name="segmentAlignmentFlag" type="xs:boolean" default="false"/>
        <xs:attribute name="bitStreamSwitchingFlag" type="xs:boolean" default="false"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <!-- RepresentationBase type; extended by other Representation-related types -->
    <xs:complexType name="RepresentationBaseType">
        <xs:sequence>
            <xs:element name="ContentProtection" type="ContentDescriptorType"
                                    minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Accessibility" type="ContentDescriptorType"
                                    minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Rating" type="ContentDescriptorType"
                                    minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Viewpoint" type="ContentDescriptorType"
                                    minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="MultipleViews" type="MultipleViewsType" minOccurs="0"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="group" type="xs:unsignedInt"/>
        <xs:attribute name="width" type="xs:unsignedInt"/>
        <xs:attribute name="height" type="xs:unsignedInt"/>
        <xs:attribute name="parx" type="xs:unsignedInt"/>
        <xs:attribute name="pary" type="xs:unsignedInt"/>
        <xs:attribute name="lang" type="LangVectorType"/>
        <xs:attribute name="mimeType" type="xs:string"/>
        <xs:attribute name="startWithRAP" type="xs:boolean"/>
        <xs:attribute name="frameRate" type="xs:double"/>
        <xs:attribute name="maximumRAPPeriod" type="xs:double"/>
        <xs:attribute name="numberOfChannels" type="StringVectorType"/>
        <xs:attribute name="samplingRate" type="StringVectorType"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <!-- Representation of the presentation content for a specific Period;
                    extends RepresentationBaseType -->
    <xs:complexType name="RepresentationType">
        <xs:complexContent>
            <xs:extension base="RepresentationBaseType">
                <xs:sequence>
                    <xs:element name="SubRepresentation" type="SubRepresentationType"
                                            minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="SegmentInfo" type="SegmentInfoType"/>
                    <xs:element name="TrickMode" type="TrickModeType" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="id" type="xs:string" use="required"/>
                <xs:attribute name="bandwidth" type="xs:unsignedInt" use="required"/>
                <xs:attribute name="qualityRanking" type="xs:unsignedInt"/>
                <xs:attribute name="dependencyId" type="StringVectorType"/>
                <xs:attribute name="bitstreamStructureId" type="StringVectorType"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- 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" use="required"/>
                <xs:attribute name="bandwidth" type="xs:unsignedInt" use="required"/>
                <xs:attribute name="trickModeApr" type="xs:double"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Group to contain information common to a group of Representations;
                    extends RepresentationBaseType -->
    <xs:complexType name="GroupType">
        <xs:complexContent>
            <xs:extension base="RepresentationBaseType">
                <xs:sequence>
                    <xs:element name="Representation" type="RepresentationType"
                                            minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="SegmentInfoDefault" type="SegmentInfoDefaultType" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute ref="xlink:href"/>
                <xs:attribute ref="xlink:actuate" default="onRequest"/>
                <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="xs:double"/>
                <xs:attribute name="maxFrameRate" type="xs:double"/>
                <xs:attribute name="subsegmentAlignment" type="xs:boolean" default="false"/>
                <xs:attribute name="segmentAlignmentFlag" type="xs:boolean"/>
                <xs:attribute name="bitStreamSwitchingFlag" type="xs:boolean"/>
                <xs:anyAttribute namespace="##other" processContents="lax"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Subset selection information on Groups -->
    <xs:complexType name="SubsetType">
        <xs:sequence>
            <xs:element name="Contains" type="ContainsType" minOccurs="1" maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <xs:complexType name="ContainsType">
        <xs:attribute name="group" type="xs:unsignedInt" use="required"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>


  <!-- Default Segment access information -->
    <xs:complexType name="SegmentInfoDefaultType">
        <xs:sequence>
            <xs:element name="InitialisationSegmentURL" type="UrlType" minOccurs="0"/>
            <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="SegmentTimeline" type="SegmentTimelineType" minOccurs="0"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attributeGroup ref="SegmentInfoAttrGroup"/>
        <xs:attribute name="sourceURLTemplatePeriod" type="xs:string"/>
        <xs:attribute name="indexTemplate" type="xs:string"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <!-- Segment access information -->
    <xs:complexType name="SegmentInfoType">
        <xs:sequence>
            <xs:element name="InitialisationSegmentURL" type="UrlType" minOccurs="0"/>
            <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="SegmentTimeline" type="SegmentTimelineType" minOccurs="0"/>
```

```
            <xs:choice minOccurs="0">
                <xs:element name="UrlTemplate" type="UrlTemplateType" minOccurs="0"/>
                <xs:sequence>
                    <xs:element name="Url" type="UrlType" maxOccurs="unbounded"/>
                    <xs:element name="Index" type="UrlType" maxOccurs="unbounded"/>
                    <xs:any namespace="##other" processContents="lax" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:element name="SegmentList" type="SegmentListType" minOccurs="0"/>
                <xs:any namespace="##other" processContents="lax" minOccurs="0"
                     maxOccurs="unbounded"/>
            </xs:choice>
        </xs:sequence>
        <xs:attributeGroup ref="SegmentInfoAttrGroup"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <!-- grouping attributes common to SegmentInfo and SegmentInfoDefault -->
    <xs:attributeGroup name="SegmentInfoAttrGroup" >
        <xs:attribute name="duration" type="xs:duration"/>
        <xs:attribute name="startIndex" type="xs:unsignedInt" default="1"/>
    </xs:attributeGroup>

    <!-- A Segment URL  -->
    <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>

    <!-- A URL template  -->
    <xs:complexType name="UrlTemplateType">
        <xs:sequence>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="sourceURL" type="xs:anyURI"/>
        <xs:attribute name="indexURL" type="xs:anyURI"/>
        <xs:attribute name="endIndex" type="xs:unsignedInt"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <!-- SegmentList allows xlink in addition to list of URLs -->
    <xs:complexType name="SegmentListType">
        <xs:sequence>
            <xs:element name="Url" type="UrlType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Index" type="UrlType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute ref="xlink:href"/>
        <xs:attribute ref="xlink:actuate" default="onRequest"/>
        <xs:attribute name="startIndex" type="xs:unsignedInt"/>
    </xs:complexType>

    <!-- Generic named descriptive information about the content -->
    <xs:complexType name="ContentDescriptorType">
        <xs:sequence>
            <xs:element minOccurs="0" name="SchemeInformation" type="xs:string"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="schemeIdUri" type="xs:anyURI" use="required"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <!-- Gives information about trick mode -->
    <xs:complexType name="TrickModeType">
        <xs:sequence>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="alternatePlayoutRate" type="xs:string"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
```

```
    <!-- Type for MultipleViews -->
    <xs:complexType name="MultipleViewsType">
        <xs:element name="FramePacking" type="ContentDescriptorType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:attribute name="stereoId" type="StringVectorType"/>
        <!-- stereoid: list of strings starting with "l" or "r" followed by optional index number -->
        <xs:attribute name="maximumViews" type="xs:unsignedInt" default="1"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <!-- Type for space delimited list of strings -->
    <xs:simpleType name="StringVectorType ">
        <xs:list itemType="xs:string"/>
    </xs:simpleType>

    <!-- Type for space delimited list of URIs -->
    <xs:simpleType name="URIVectorType ">
        <xs:list itemType="xs:anyURI"/>
    </xs:simpleType>

  <!-- Type for space delimited list of language codes -->
    <xs:simpleType name="LangVectorType">
        <xs:list itemType="xs:language"/>
    </xs:simpleType>

    <!-- Supplementary URL to the one given as attribute -->
    <xs:complexType name="BaseURLType">
        <xs:simpleContent>
                <xs:extension base="xs:anyURI">
                    <xs:anyAttribute namespace="##other" processContents="lax"/>
                </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

</xs:schema>
```

Annex D
(normative)

MIME Type Registration for MPD

D.1 Introduction

This Annex provides the formal MIME type registration for the MPD. It is referenced from the registry at http://www.iana.org/.

D.2 MIME Type and Subtype

The MIME Type and Subtype are defined as follows:

— Media Type Name: video <<ed: seems like we should think about "application" here>>

— Subtype name: vnd.mpeg.dash.mpd

— Required parameters: none

— Optional parameters: The ‘profiles’ parameter as documented in the published specification

— Encoding considerations: 8 bit text; the MPD is an XML document

— Security considerations: The MPD is a media presentation description and contains references to other resources. It is coded in XML, and there are risks that deliberately malformed XML could cause security issues. In addition, an MPD could be authored that causes receiving clients to access other resources; if widely distributed, this could be used to cause a denial-of-service attack.

— Interoperability considerations:
The specification defines a platform-independent expression of a presentation, and it is intended that wide interoperability can be achieved.

— Published specification: ISO/IEC 23001-6: Information technology — MPEG systems technologies — Part 6: Dynamic adaptive streaming over HTTP (DASH)

— Applications which use this media type: various

— Additional information:

  — Magic number(s): none

  — File extension(s): mpd

  — Macintosh File Type Code(s): n/a

  — Object Identifier(s) or OID(s): none

  — Intended usage: common

— Other Information/General Comment: none

— Person to contact for further information:

  — Name: Thomes Stockhammer

  — Email: stockhammer@nomor.de

— Author/Change controller: ISO/IEC JTC1/SC29 (MPEG)

D.3 Parameters

D.3.1 The profiles parameter

Parameter name: Profiles

Parameter value: A single value, or a comma-separated list of values identifying the profiles(s) to which the file claims conformance.

Note that, per RFC 2045, some characters (including the comma used to separate multiple values) require that the entire parameter value be enclosed in quotes.

An element may include an octet that must be encoded in order to comply with RFC 2045. In this case, RFC 2231 is used: an asterisk ("*") is placed at the end of the parameter name (becoming “profiles*” instead of “profiles”), the parameter value usually starts with two single quote (“'”) characters (indicating that neither character set nor language is specified), and each octet that requires encoding is represented as a percent sign (“%”) followed by two hexadecimal digits. Note that, when the RFC 2231 form is used, the percent sign, asterisk, and single quote characters have special meaning and so must themselves be encoded.

The 'profiles' parameter is an optional parameter that indicates one or more profiles to which the file claims conformance. The value is a space-delimited list of profile identifiers. The profile identifiers reported in the MIME type parameter should match identically the profiles reported in the profiles attribute in the MPD itself (see Clause 8).

example:

```
video/vnd.mpeg.dash.mpd;profiles="urn:mpeg:mpegB:profiles:dash:mythical
   urn:3gpp:dash:profiles:fictional"
```

Annex E
(normative)

DASH Quality Metrics

E.1 Introduction

This annex defines the ISO/IEC 23001-6 quality metrics (DASH-QM).

E.2 DASH-QM client reference model

The DASH-QM client reference model is depicted in highlighting so-called observation points (OPs) as defined in E.3.

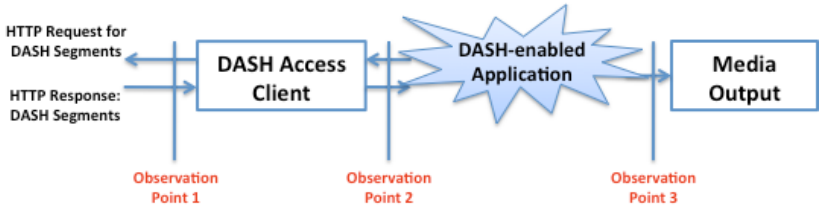

Figure E.1 — DASH-QM client reference model

The *DASH access client* as defined in 4.2 in this part of ISO/IEC 23001, issues HTTP requests (for DASH data structures), and receives HTTP request responses (containing DASH data structures). Data structures may typically be PDs, segments or partial segments. This input/output interface from the network towards the DASH client is referred to as observation point 1 (OP1).

Furthermore, the DASH client delivers encoded media samples to the *DASH-enabled application* for further processing and may receive also commands from it. This input/output interface of the DASH client towards the DASH-enabled application is referred to as observation point 2 (OP2).

EXAMPLE Further processing may include de-multiplexing (of audio/video) and/or decoding potentially involving several buffers.

Finally, the DASH-enabled application delivers decoded media samples to the *media output*, which displays the media to the user. This output interface towards the user is referred to as observation point 3 (OP3).

E.3 Definition of observation points

E.3.1 Introduction

This Clause defines the observation points as depicted in Figure E.1.

E.3.2 Observation point 1

The observation point 1 (OP1) is defined as:

— a set of TCP connections each defined by its destination IP address, initiation, connect and close times;

— a sequence of transmitted HTTP requests, each defined by its transmission time, contents, and the TCP connection on which it is sent; and

— for each HTTP response, the reception time and contents of the response header and the reception time of each byte of the response body.

NOTE The contents of the response body is fully defined by the contents of the request and response headers.

E.3.3 Observation point 2

The observation point 2 (OP2) consists of encoded media samples. Each encoded media sample is defined as:

— media type;

— decoding time;

— presentation time;

— the `@id` of the Representation from which the sample is taken; and

— the delivery time.

E.3.4 Observation point 3

The observation point 3 (OP3) consists of decoded media samples. Each decoded media sample is defined as:

— the media type;

— the presentation timestamp of the sample (media time);

— the actual presentation time of the sample (real time); and

— the `@id` of the Representation from which the sample is taken (the highest dependency level if the sample was constructed from multiple Representations).

E.4 Semantics of the quality metrics

E.4.1 Introduction

This Clause defines the semantics of the quality metrics at each observation point. The semantics are defined using an abstract syntax, which can be easily mapped to the concrete syntax depending on the application-specific needs (e.g., XML, JSON, Common Log format). 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*.

E.4.2 Observation point 1 (OP1)

E.4.2.1 General metrics

For each TCP connection:

Key	Type	Description
dest	String	IP Address of the destination.
topen	Real Time	The time at which the connection was opened (sending time of the initial SYN).
tclose	Real Time	The time at which the connection was closed (sending or reception time of FIN or RST).
tconnect	Integer	Connect time in ms (time from sending the initial SYN to receiving the ACK).
id	Integer	An identifier for the connection which is unique during the lifetime of the connection.

For each HTTP request and, thus, also TCP connection:

Key	Type	Description
tcpid	Integer	Identifier of the TCP connection on which the HTTP request was sent.
url	String	The URL requested.
range	String	The contents 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).
trace	List	Throughput trace.
entry	Objects	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.

The periods reported in `tpentry` 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).

E.4.2.2 Derived metrics

Furthermore, the following DASH QM-related metrics may be derived from the information described in E.4.2.1:

Key	Type	Description
`atdtrace`	List	Arrival time deviation trace.
`entry`	Objects	A single arrival time deviation measurement entry.
`t`	Real Time	Measurement time.
`rep`	String	Representation id at measurement time.
`id`	Integer	Segment id at measurement time.
`byte`	Integer	Byte offset of byte chosen for this measurement entry
`m`	Float	The time (i.e., delta) between the data arrival time required for maintaining constant buffer size at the client and the actual arrival time of a byte which is denoted as *atd_m* and defined in Equation (1). $tres_i$ corresponds to the arrival time of the *i*th byte, and t_x describes the presentation time of the sample containing byte x. $$atd\_m = tres_i - tres_0 - t_x \quad (1)$$ NOTE 1 This metric can be calculated for any byte to the granularity of `tpinterval` from `tptrace`. The value may be reported, for example, for the first byte of each HTTP request. NOTE 2 It can be considered something like the network jitter (or delay variation) but at the level of the DASH layer.
`tpvartrace`	List	Throughput variation trace.
`count`	Integer	The throughput measurement interval in units of `tpinterval`
`entry`	Objects	A single throughput variation measurement entry.
`s`	Real Time	Measurement period start.
`d`	Integer	Measurement period duration (ms).
`a`	Float	The average throughput during the measurement period in bits/s.

		v	Float	The variance of the throughput values for every contiguous group of count measurement intervals in $(bits/s)^2$

E.4.3 Observation point 2 (OP2)

Encoded Samples are typically stored in a media buffer to compensate jitter in the delivery variable bitrate coding, and composition offsets. Underruns in the encoded sample buffer typically result in display problems. Suitable reporting at OP2 describes each event for which data is written in the encoded sample buffer.

Key	Type	Description
start	Real Time	Timestamp of the action.
periodid	Integer	The id of the period from which the bytes are taken.
representationid	Integer	The id of the representation from which the bytes are taken.
segmentindex	Integer	The index of the segment from which the bytes are taken.
byteindex	Integer	Index for the first requested byte in the segment.
size	Integer	Size of continuous bytes written for this action.
buffersize	Integer	The number of bytes in the buffer after the new bytes were written.

E.4.4 Observation point 3 (OP3)

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 at OP3 can thus be constructed from a list of time periods during which samples of a single representation were continuously delivered, such that each was presented at its specified presentation time to some specific level of accuracy (e.g., +/-10ms).

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.

Key	Type	Description
start	Real Time	Timestamp of the user action which triggered playout.
mstart	Media Time	The presentation time at which playout was requested by the user action.
stopreason	Enum	Reason for stopping: — User Request — End of Content — Failure.

trace		List	List of periods of continuous rendering of decoded samples.
	traceentry	Objects	Single entry in the list.
		representationid	String	The id of the representation from which the samples were taken.
		start	Real Time	The time at which the first sample was rendered.
		mstart	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.

The above captures a complete record of what the user saw from one user action to the next, assuming only that each sample that was received was correctly rendered.

Annex F
(informative)

MPD Examples and MPD Usage

F.1 Example 1: On-Demand

Simple example for a piece of On-Demand content, with self-initializing media segments, multiple languages, subtitles, content protection and multiple base URLs.

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD
    xmlns:xsi="http://www.w3.org/2001/XMLSchema"
    xmlns="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"
    xsi:schemaLocation="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"
    type="OnDemand"
    mediaPresentationDuration="PT3256S"
    minBufferTime="PT1.2S"
    profiles="urn:mpeg:mpegB:profile:dash:isoff-basic-on-demand">

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

    <Period>
        <Group mimeType="audio/mp4; codecs=mp4a.0x40" lang="en" subsegmentAlignment="true">
            <ContentProtection schemeIdUri="urn:uuid:706D6953-656C-5244-4D48-656164657221"/>
            <Representation bandwidth="64000">
                <SegmentInfo>
                    <BaseURL>7657412348.mp4</BaseURL>
                </SegmentInfo>
            </Representation>
            <Representation bandwidth="32000">
                <SegmentInfo>
                    <BaseURL>3463646346.mp4</BaseURL>
                </SegmentInfo>
            </Representation>
        </Group>

        <Group mimeType="audio/mp4; codec=mp4a.40.2" lang="fr" subsegmentAlignment="true">
            <ContentProtection schemeIdUri="urn:uuid:706D6953-656C-5244-4D48-656164657221"/>
            <Representation bandwidth="64000">
                <SegmentInfo>
                    <BaseURL>3463275477.mp4</BaseURL>
                </SegmentInfo>
            </Representation>
            <Representation bandwidth="32000">
                <SegmentInfo>
                    <BaseURL>5685763463.mp4</BaseURL>
                </SegmentInfo>
            </Representation>
        </Group>

        <Group mimeType="application/ttml+xml" lang="de">
            <Representation bandwidth="256">
                <SegmentInfo>
                    <BaseURL>796735657.mp4</BaseURL>
                </SegmentInfo>
            </Representation>
        </Group>

        <Group mimeType="video/mp4; codecs=avc1.4d0228" subsegmentAlignment="true">
            <ContentProtection schemeIdUri="urn:uuid:706D6953-656C-5244-4D48-656164657221"/>
            <Representation bandwidth="256000" width="320" height="240">
                <SegmentInfo>
                    <BaseURL>8563456473.mp4</BaseURL>
```

```
                </SegmentInfo>
            </Representation>
            <Representation bandwidth="512000" width="320" height="240">
                <SegmentInfo>
                    <BaseURL>56363634.mp4</BaseURL>
                </SegmentInfo>
            </Representation>
            <Representation bandwidth="1024000" width="640" height="480">
                <SegmentInfo>
                    <BaseURL>562465736.mp4</BaseURL>
            </SegmentInfo>
            </Representation>
            <Representation bandwidth="1384000" width="640" height="480">
                <SegmentInfo>
                    <BaseURL>41325645.mp4</BaseURL>
                </SegmentInfo>
            </Representation>
            <Representation bandwidth="1536000" width="1280" height="720">
                <SegmentInfo>
                    <BaseURL>89045625.mp4</BaseURL>
                </SegmentInfo>
            </Representation>
            <Representation bandwidth="2048000" width="1280" height="720">
                <SegmentInfo>
                    <BaseURL>23536745734.mp4</BaseURL>
                </SegmentInfo>
            </Representation>
        </Group>
    </Period>
</MPD>
```

F.2 Example 2: Multiple Views

Simple example for a piece of multiple view content.

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD
    xmlns:xsi="http://www.w3.org/2001/XMLSchema"
    xmlns="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"
    xsi:schemaLocation="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"
    profiles= "urn:mpeg:mpegB:profile:dash:full:2011"
    minBufferTime="PT10.00S"
    <BaseURL>http://www.example.com/</BaseURL>
    <Period start="PT0.00S" duration="PT30.00S">
    <!--  In this Period there are 3 views: coming from three lined up cameras: C1-C2-C3.
       C1+C2 form a stereo pair and C2+C3, too. But C1+C3 do not.
       C2 is taken as base view for MVC; C1 and C3 are enhancement views -->
        <Group mimeType="video/mp4; codecs=avc1.640828" bandwidth="128000">
            <Representation id="C2">
                <MultipleViews stereoId="r1 l2"/>
                <SegmentInfo duration="PT10.00S">
                    <InitialisationSegmentURL sourceURL="seg-m-init.mp4"/>
                    <Url sourceURL="seg-m1-C2view-1.mp4"/>
                    <Url sourceURL="seg-m1-C2view-2.mp4"/>
                    <Url sourceURL="seg-m1-C2view-3.mp4"/>
                </SegmentInfo>
            </Representation>
        </Group>
        <Group mimeType="video/mp4; codecs=mvc1.760028" bandwidth="64000">
            <Representation id="C1" dependencyId="C2">
                <MultipleViews stereoId="l1"/>
                <SegmentInfo duration="PT10.00S">
                    <InitialisationSegmentURL sourceURL="seg-m-init.mp4"/>
```

```
                    <Url sourceURL="seg-m1-C1view-1.mp4"/>
                    <Url sourceURL="seg-m1-C1view-2.mp4"/>
                    <Url sourceURL="seg-m1-C1view-3.mp4"/>
                </SegmentInfo>
            </Representation>
            <Representation id="C3" dependencyId="C2">
                <MultipleViews stereoId="r2"/>
                    <SegmentInfo duration="PT10.00S">
                    <InitialisationSegmentURL sourceURL="seg-m-init.mp4"/>
                    <Url sourceURL="seg-m1-C3view-1.mp4"/>
                    <Url sourceURL="seg-m1-C3view-2.mp4"/>
                    <Url sourceURL="seg-m1-C3view-3.mp4"/>
                </SegmentInfo>
            </Representation>
        </Group>
    </Period>
    <Period duration="PT20.00S">
    <!--  In this Period there are only 2 views:
       C1+C2 form a stereo pair;
       C2 is the base view for MVC and C1 is the enhancement view -->
        <Group mimeType="video/mp4; codecs=avc1.640828" bandwidth="128000">
            <Representation id="C2">
                <MultipleViews stereoId="r"/>
                <SegmentInfo duration="PT10.00S">
                    <InitialisationSegmentURL sourceURL="seg-m-init.mp4"/>
                    <Url sourceURL="seg-m1-C2view-4.mp4"/>
                    <Url sourceURL="seg-m1-C2view-5.mp4"/>
                </SegmentInfo>
            </Representation>
        </Group>
        <Group mimeType="video/mp4; codecs=mvc1.760028" bandwidth="64000">
            <Representation id="C1" dependencyId="C2">
                <MultipleViews stereoId="l"/>
                <SegmentInfo duration="PT10.00S">
                    <InitialisationSegmentURL sourceURL="seg-m-init.mp4"/>
                    <Url sourceURL="seg-m1-C1view-4.mp4"/>
                    <Url sourceURL="seg-m1-C1view-5.mp4"/>
                </SegmentInfo>
            </Representation>
        </Group>
    </Period>
</MPD>
```

F.3 Example 3: SVC views

Simple example for a piece of SVC content.

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD
    xmlns:xsi="http://www.w3.org/2001/XMLSchema"
    xmlns="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"
    xsi:schemaLocation="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"
    profiles= "urn:mpeg:mpegB:profile:dash:full:2011"
    minBufferTime="PT10.00S"
    <BaseURL>http://www.example.com/</BaseURL>
    <Period start="PT0S">
```

```
    <!--  In this Period there are 2 alternative SVC streams:
       An SVC stream with a base layer representation and a complementary enhancement layer representation
       and another SVC with a base layer and two complementary enhancement representations -->
        <Group mimeType="video/mp4; codecs=avc1.640028">
            <Representation width="320" height="240" frameRate="15" id="tag0" bandwidth="128000">
                <SegmentInfo duration="PT10.00S">
                    <InitialisationSegmentURL sourceURL="seg-s-init.mp4"/>
                    <Url sourceURL="seg-s1-128k-1.mp4"/>
                    <Url sourceURL="seg-s1-128k-2.mp4"/>
                    <Url sourceURL="seg-s1-128k-3.mp4"/>
                </SegmentInfo>
            </Representation>
            <Representation width="1280" height="720" frameRate="30" id="tag2" bandwidth="2560000">
                <SegmentInfo duration="PT10.00S">
                    <InitialisationSegmentURL sourceURL=" seg-s-init.mp4"/>
                    <Url sourceURL="seg-s2-2560k-1.mp4"/>
                    <Url sourceURL="seg-s2-2560k-2.mp4"/>
                    <Url sourceURL="seg-s2-2560k-3.mp4"/>
                </SegmentInfo>
            </Representation>
        </Group>
        <Group mimeType="video/mp4; codecs=avc2.560028">
            <Representation width="320" height="240" frameRate="15" id="tag1" dependencyId="tag0"
bandwidth="128000">
                <SegmentInfo duration="PT10.00S">
                    <InitialisationSegmentURL sourceURL="seg-s-init.mp4"/>
                    <Url sourceURL="seg-s1-256k-1.mp4"/>
                    <Url sourceURL="seg-s1-256k-2.mp4"/>
                    <Url sourceURL="seg-s1-256k-3.mp4"/>
                </SegmentInfo>
            </Representation>
            <Representation width="1280" height="720" frameRate="60" id="tag3" dependencyId="tag2"
bandwidth="2560000">
                <SegmentInfo duration="PT10.00S">
                    <InitialisationSegmentURL sourceURL="seg-s-init.mp4"/>
                    <Url sourceURL="seg-s2-5120k-1.mp4"/>
                    <Url sourceURL="seg-s2-5120k-2.mp4"/>
                    <Url sourceURL="seg-s2-5120k-3.mp4"/>
                </SegmentInfo>
            </Representation>
        </Group>
        <Group mimeType="video/mp4; codecs=avc2.560028" width="1920" height="1080" frameRate="60"
bandwidth="5120000">
            <Representation id="tag4" dependencyId="tag2 tag3">
                <SegmentInfo duration="PT10.00S">
                    <InitialisationSegmentURL sourceURL="seg-s-init.mp4"/>
                    <Url sourceURL="seg-s2-10240k-1.mp4"/>
                    <Url sourceURL="seg-s2-10240k-2.mp4"/>
                    <Url sourceURL="seg-s2-10240k-3.mp4"/>
                </SegmentInfo>
            </Representation>
        </Group>
        <Group mimeType="audio/mp4; codecs=mp4a.40.2" bandwidth="64000">
            <Representation id="tag5">
                <SegmentInfo duration="PT10.00S">
                    <InitialisationSegmentURL sourceURL="seg-s-init.mp4"/>
                    <Url sourceURL="seg-a1-64k-1.mp4"/>
                    <Url sourceURL="seg-a1-64k-2.mp4"/>
                    <Url sourceURL="seg-a1-64k-3.mp4"/>
                </SegmentInfo>
```

```
            </Representation>
        </Group>
    </Period>
</MPD>
```

F.4 Example 4: Content Protected by Multiple Schemes

In the example below, *example.com* is a provider of CDN services and also a hosting service for movie service providers *MoviesSP*. The English audio and the video tracks are encrypted and licensed by MoviesSP. However, the French audio track is encrypted and licensed by a different service provider.

A hypothetical DRM standardization organization has registered a Scheme Type 'ZZZZ' with MP4REG and documented how scheme specific licensing information is stored entirely within the content so there is no **`SchemeInformation`** element in the **`ContentProtection`** element. Since the Scheme Type is registered and the rules for its use are documented, the "`urn:mpeg:mpegB:dash:mp4protection:ZZZZ`" `@schemeIdUri` is used.

In addition, a second DRM scheme being used comes from a DRM vendor who has published documentation of their system that declares they use the DASH **`ContentProtection`** element with a `@schemeIdUri` attribute value "`http://example.net/052011/drm`". (This DRM vendor owned the domain *example.net* as of May, 2011.) Documentation for this scheme states there must always be two URLs in the **`SchemeInformation`** element. The first is always for a license token and the second is always for a content token. Regardless of which service provider uses the protection product from this DRM vendor, these rules must always be followed, so the meaning of the **`SchemeInformation`** and how it is parsed is always the same.

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD
    xmlns:xsi="http://www.w3.org/2001/XMLSchema"
    xmlns="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"
    xsi:schemaLocation="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"
    minBufferTime="PT10.00S"
    type="OnDemand"
    mediaPresentationDuration="PT3256S"
    profiles="urn:mpeg:mpegB:profile:dash:isoff-basic-on-demand">

    <BaseURL>http://cdn.example.com/movie23453235/</BaseURL>

    <Period>
        <Group mimeType="audio/mp4; codec=mp4a.0x40" lang="en">
            <ContentProtection schemeIdUri="http://example.net/052011/drm">
                <SchemeInformation>
                    http://MoviesSP.example.com/protect?license=kljklsdfiowek
                    http://MoviesSP.example.com/protect?content=oyfYvpo8yFyvyo8f
                </SchemeInformation>
            </ContentProtection>
            <Representation id="1" bandwidth="64000">
                <SegmentInfo>
                    <BaseURL>audio/en/64.mp4</BaseURL>
                </SegmentInfo>
            </Representation>
        </Group>

        <Group mimeType="audio/mp4; codec=mp4a.0x40" lang="fr">
            <ContentProtection schemeIdUri=" urn:mpeg:mpegB:dash:mp4protection:ZZZZ"/>
            <Representation id="3" bandwidth="64000">
                <SegmentInfo>
                    <BaseURL>audio/fr/64.mp4</BaseURL>
                </SegmentInfo>
            </Representation>
        </Group>

        <Group mimeType="application/ttml+xml" lang="de">
            <Representation id="5" bandwidth="256">
```

```
                <SegmentInfo>
                    <BaseURL>subtitles/de.xml</BaseURL>
                </SegmentInfo>
            </Representation>
        </Group>

        <Group mimeType="video/mp4; codec=avc1" subsegmentAlignment="true">
            <ContentProtection schemeIdUri="http://example.net/052011/drm">
                <SchemeInformation>
                    http://MoviesSP.example.com/protect?license=jfjhwlsdkfiowkl
                    http://MoviesSP.example.com/protect?content=mslkfjsfiowelkfl
                </SchemeInformation>
            </ContentProtection>
            <Representation id="6" bandwidth="256000" width="320" height="240">
                <SegmentInfo>
                    <BaseURL>video256.mp4</BaseURL>
                </SegmentInfo>
            </Representation>
            <Representation id="7" bandwidth="512000" width="320" height="240">
                <SegmentInfo>
                    <BaseURL>video512.mp4</BaseURL>
                </SegmentInfo>
            </Representation>
            <Representation id="8" bandwidth="1024000" width="640" height="480">
                <SegmentInfo>
                    <BaseURL>video1024.mp4</BaseURL>
                </SegmentInfo>
            </Representation>
            <SegmentInfoDefault>
                <BaseURL>video/</BaseURL>
            </SegmentInfoDefault>
        </Group>
    </Period>
</MPD>
```

F.5 Usage of MPD Assembly

F.5.1 EXAMPLE 1: Group Referencing

```
...
<Group xlink:href="http://server.com/repgr-bw244k-bw312k.xml"
                    xlink:actuate="onRequest" minBW="249967" maxBW="319488"
                    mimeType="video/3gpp"/>
...
```

The xml document with URL `http://server.com/repgr-bw244k-bw312k.xml` may look as follows:

```
<Group minBW="249967" maxBW="319488" mimeType="video/3gpp"
                    group="3"/>
<Representation group="3" bandwidth="249967">
  <SegmentInfo duration="PT10.00S">
    <InitialisationSegmentURL sourceURL="http://server.com/gr3/init.3gp"/>
    <Url sourceURL="http://server.com/gr3/seg_1.3gs" ext:start="PT0.00S" />
    <Url sourceURL="http://server.com/gr3/seg_2.3gs" ext:start="PT9.68S" />
    <Url sourceURL="http://server.com/gr3/seg_3.3gs" ext:start="PT19.64S" />
    <Url sourceURL="http://server.com/gr3/seg_4.3gs" ext:start="PT29.68S" />
    <!-- etc. -->
  </SegmentInfo>
</Representation>
<Representation group="3" bandwidth="319488">
  <SegmentInfo duration="PT10.00S">
    <InitialisationSegmentURL sourceURL="http://server.com/gr3/init.3gp"/>
    <Url sourceURL="http://server.com/gr3/seg_1.3gs" ext:start="PT0.00S" />
    <Url sourceURL="http://server.com/gr3/seg_2.3gs" ext:start="PT9.68S" />
    <Url sourceURL="http://server.com/gr3/seg_3.3gs" ext:start="PT19.64S" />
```

```
    <Url sourceURL="http://server.com/gr3/seg_4.3gs" ext:start="PT29.68S" />
    <!-- etc. -->
  </SegmentInfo>
</Representation>
```

After XLink processing which – in this case – is triggered on request by the application, the following shall be available to the application which is expressed in XML for better presentation purposes. In an application, this would be probably available in DOM nodes, SAX events, or similar.

```
<Group minBW="249967" maxBW="319488" mimeType="video/3gpp"
                    group="3"/>
<Representation group="3" bandwidth="249967">
  <SegmentInfo duration="PT10.00S">
    <InitialisationSegmentURL sourceURL="http://server.com/gr3/init.3gp"/>
    <Url sourceURL="http://server.com/gr3/seg_1.3gs" ext:start="PT0.00S" />
    <Url sourceURL="http://server.com/gr3/seg_2.3gs" ext:start="PT9.68S" />
    <Url sourceURL="http://server.com/gr3/seg_3.3gs" ext:start="PT19.64S" />
    <Url sourceURL="http://server.com/gr3/seg_4.3gs" ext:start="PT29.68S" />
    <!-- etc. -->
  </SegmentInfo>
</Representation>
<Representation group="3" bandwidth="319488">
  <SegmentInfo duration="PT10.00S">
    <InitialisationSegmentURL sourceURL="http://server.com/gr3/init.3gp"/>
    <Url sourceURL="http://server.com/gr3/seg_1.3gs" ext:start="PT0.00S" />
    <Url sourceURL="http://server.com/gr3/seg_2.3gs" ext:start="PT9.68S" />
    <Url sourceURL="http://server.com/gr3/seg_3.3gs" ext:start="PT19.64S" />
    <Url sourceURL="http://server.com/gr3/seg_4.3gs" ext:start="PT29.68S" />
    <!-- etc. -->
  </SegmentInfo>
</Representation>
```

F.5.2 EXAMPLE 2: Period Referencing

The following example shows a Period element including `xlink:href` and `xlink:actuate` attributes among others.

```
...
<Period xlink:href="http://example.com/Period-06.xml"
              xlink:actuate="onRequest" start="PT06H"/>
...
```

The xml document with URL `http://example.com/Period-06.xml` may look as follows:

```
<Period start="PT06H" id=”10” segmentAlignmentFlag=”true”>
        <SegmentInfoDefault duration="PT10S" sourceURLTemplatePeriod="http://www.example.com/Period-
06/rep-$RepresentationID$/seg-$Index$.3gp"/>
        <Representation id="QVGA-LQ" mimeType='video/3gpp; codecs="avc1.42E00C, mp4a.40.2"'
bandwidth="192000" width="320" height="240">
            <SegmentInfo>
                <InitialisationSegmentURL sourceURL="http://www.example.com/rep-QVGA-LQ/seg-
init.3gp"/>
            </SegmentInfo>
        </Representation>
        <Representation id="QVGA-HQ" mimeType='video/3gpp; codecs="avc1.42E00C, mp4a.40.2"'
bandwidth="384000" width="320" height="240">
            <SegmentInfo>
                <InitialisationSegmentURL sourceURL="http://www.example.com/rep-QVGA-HQ/seg-
init.3gp"/>
            </SegmentInfo>
        </Representation>
</Period>
```

After XLink processing which – in this case – is triggered on request by the application, the following shall be available to the application which is expressed in XML for better presentation purposes. In an application, this would be probably available in DOM nodes, SAX events, or similar.

```
<Period start="PT06H" id=”10” segmentAlignmentFlag=”True”>
        <SegmentInfoDefault  duration="PT10S" sourceURLTemplatePeriod="http://www.example.com/Period-
06/rep-$RepresentationID$/seg-$Index$.3gp"/>
        <Representation id="QVGA-LQ" mimeType='video/3gpp; codecs="avc1.42E00C, mp4a.40.2"'
bandwidth="192000" width="320" height="240">
            <SegmentInfo>
                <InitialisationSegmentURL sourceURL="http://www.example.com/rep-QVGA-LQ/seg-
init.3gp"/>
            </SegmentInfo>
        </Representation>
        <Representation id="QVGA-HQ" mimeType='video/3gpp; codecs="avc1.42E00C, mp4a.40.2"'
bandwidth="384000" width="320" height="240">
            <SegmentInfo>
                <InitialisationSegmentURL sourceURL="http://www.example.com/rep-QVGA-HQ/seg-
init.3gp"/>
            </SegmentInfo>
        </Representation>
</Period>
```

Bibliography

[1] 3GPP TS 26.234, *Transparent end-to-end packet switched streaming service (PSS); Protocols and codecs,* http://www.3gpp.org/ftp/Specs/html-info/26234.htm

[2] 3GPP TS 26.244, *Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP)*, http://www.3gpp.org/ftp/Specs/html-info/26244.htm

[3] IETF RFC 1952, *GZIP file format specification version 4.3*, May 1996

[4] IETF RFC 2231, *MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations*, November 1997

[5] IETF RFC 2405, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, November 1996

[6] IETF RFC 2818, *HTTP Over TLS*, May 2000

[7] IETF RFC 5905, *Network Time Protocol Version 4: Protocol and Algorithms Specification*, June 2010