INTERNATIONAL STANDARD ISO/IEC 23009-1:2012
TECHNICAL CORRIGENDUM 1

Published 2013-06-01

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

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

TECHNICAL CORRIGENDUM 1

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

Partie 1: Description de la présentation et formats de remise des médias

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to ISO/IEC 23009-1:2012 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information.*

Clause 2

Replace:

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

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

ICS 35.040 **Ref. No. ISO/IEC 23009-1:2012/Cor.1:2013(E)**

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)*

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

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

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

IETF RFC 2141, *URN Syntax*, May 1997

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

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

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

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

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

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

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

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

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

with:

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

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)*

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

IETF RFC 2141, *URN Syntax*, May 1997

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

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

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

IETF RFC 3629, *UTF-8, a transformation format of ISO 10646*, November 2003
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 4288, *Media Type Specifications and Registration Procedures*, December 2005

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

IETF RFC 4648, *The Base16, Base32, and Base64 Data Encodings*, October 2006
IETF RFC 5646, *Tags for Identifying Languages*, September 2009

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

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

3.1.29

Replace:

remote element
element that is not fullly contained in the MPD document but is referenced in the MPD with an HTTP-URL

with:

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

3.3

Replace:

— Namespace qualification of elements and attributes is used as per XML standards, in the form of **`namespace:Element`** or `@namespace:attribute` The fully qualified namespace will be provided in the schema fragment associated with the declaration.

with:

— Namespace qualification of elements and attributes is used as per XML standards, in the form of **`namespace:Element`** or `@namespace:attribute` The fully qualified namespace will be provided in the schema fragment associated with the declaration. External specifications extending the namespace of DASH are expected to document the element name in the semantic table with an extension namespace prefix.

4.3 DASH data model overview, paragraphs 12 to 14

Replace:

Segments may be further subdivided into **Subsegments** each of which contains a whole number of complete access units. There may also be media-format-specific restrictions on Subsegment boundaries, for example in the ISO Base Media File Format a Subsegment must contain a whole number of complete movie fragments. If a Segment is divided into Subsegments are described by a compact **Segment index**, which provides the presentation time range in the Representation and corresponding byte range in the Segment occupied by each Subsegment. Clients may download this index in advance and then issue requests for individual Subsegments.

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

Segmentation and Subsegmentation may be performed in ways that make switching simpler. For example, in the very simplest cases each Segment or Subsegment begins with a random access point and the boundaries of Segments or Subsegments are aligned across the Representations of an Adaptation Set. In this case, switching Representation involves playing to the end of a (Sub)Segment of one Representation and then playing from the beginning of the next (Sub)Segment of the new Representation. The Media Presentation Description and Segment Index provide various indications, which describe properties of the Representations that may make switching simpler. Profiles of this specification may then require these indicators to be set in certain ways, making implementation of clients for those profiles simpler at the cost of requiring the media data to obey the indicated constraints.

with:

Segments may be further subdivided into **Subsegments** each of which contains a whole number of complete access units. There may also be media-format-specific restrictions on Subsegment boundaries, for example in the ISO Base Media File Format a Subsegment must contain a whole number of complete movie fragments. If a Segment is divided into Subsegments they are described by a compact **Segment index**, which provides the presentation time range in the Representation and corresponding byte range in the Segment occupied by each Subsegment. Clients may download this index in advance and then issue requests for individual Subsegments.

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

Segmentation and Subsegmentation may be performed in ways that make switching simpler. For example, in the very simplest cases each Segment or Subsegment begins with a stream access point and the boundaries of Segments or Subsegments are aligned across the Representations of an Adaptation Set. In this case, switching Representation involves playing to the end of a (Sub)Segment of one Representation and then playing from the beginning of the next (Sub)Segment of the new Representation. The Media Presentation Description and Segment Index provide various indications, which describe properties of the Representations that may make switching simpler. Profiles of this specification may then require these indicators to be set in certain ways, making implementation of clients for those profiles simpler at the cost of requiring the media data to obey the indicated constraints.

4.6 Brands

Replace:

`sims`	6.3.4.4	Media Segment conforming to the Sub-Indexed Media Segment format type for ISO base media file format.
`dash`	6.3.5.2	ISO base media file format file specifically designed for DASH including movie fragments and Segment Index.

with:

`sims`	6.3.4.4	Media Segment conforming to the Sub-Indexed Media Segment format type for ISO base media file format.
`dsms`	6.3.5.1	Media Segment conforming to the DASH Self-Initializing Media Segment format type for ISO base media file format.
`dash`	6.3.5.2	ISO base media file format file specifically designed for DASH including movie fragments and Segment Index.

4.7 Schemes

Replace:

Table 2 — Schemes defined in this part of ISO/IEC 23009

Scheme Identifier	**Clause in this part of ISO/IEC 23009**	**Informative description**
`urn:mpeg:dash:mpd:2011`	Annex B	The namespace of the XML schema for the MPD.
`urn:mpeg:dash:14496:10:frame_packing_arrangement_type:2011`	5.8.5.3	frame-packing arrangement as defined by Table D-8 of ISO/IEC 14496-10.
`urn:mpeg:dash:13818:1:stereo_video_format_type:2011`	5.8.5.3	frame-packing arrangement as defined by Table L-1 of ISO/IEC 13818-1.
`urn:mpeg:dash:23003:3:audio_channel_configuration:2011`	5.8.5.4	channel configuration as defined by Table 65 of ISO/IEC 23003-3.

with:

Table 2 — Schemes defined in this part of ISO/IEC 23009

Scheme Identifier	**Clause in this part of ISO/IEC 23009**	**Informative description**
`urn:mpeg:dash:schema:mpd:2011`	Annex B	The namespace of the XML schema for the MPD.
`urn:mpeg:dash:14496:10:frame_packing_arrangement_type:2011`	5.8.5.3	frame-packing arrangement identifier
`urn:mpeg:dash:13818:1:stereo_video_format_type:2011`	5.8.5.3	frame-packing arrangement identifier
`urn:mpeg:dash:23003:3:audio_channel_configuration:2011`	5.8.5.4	channel configuration identifier

5.2.1 General

Replace:

The MIME type of the MPD document is defined in Annex C.

with:

The MIME type of the MPD document is defined in Annex C.

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

5.2.2 Schema

Replace:

```
<?xml version="1.0"?>
<xs:schema targetNamespace="urn:mpeg:DASH:schema:MPD:2011"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns="urn:mpeg:DASH:schema:MPD:2011">
  <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 the Media Presentation Description for MPEG-DASH.
    </xs:documentation>
  </xs:annotation>

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

  ...

</xs:schema>
```

with:

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

  <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 the Media Presentation Description for MPEG-DASH.
    </xs:documentation>
  </xs:annotation>

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

  ...

</xs:schema>
```

5.3.1.1 Overview

Remove the fourth list point:

— A Representation may contain one or more Sub-Representations as described in 5.3.6.

5.3.1.2 Semantics

Replace:

`@timeShiftBufferDepth`	O	specifies the duration of the time shifting buffer that is guaranteed to be available for a Media Presentation with type `'dynamic'`. When not present, the value is infinite. This value of the attribute is undefined if the type attribute is equal to `'static'`.

with:

`@timeShiftBufferDepth`	O	specifies the duration of the smallest time shifting buffer for any Representation in the MPD that is guaranteed to be available for a Media Presentation with type `'dynamic'`. When not present, the value is infinite. This value of the attribute is undefined if the type attribute is equal to `'static'`.

5.3.2.2 Semantics

Replace:

Table 4 — Semantics of `Period` element

Element or Attribute Name	Use	Description
`Period`		specifies the information of a Period.
`@xlink:href`	O	specifies a reference to an external **`Period`** element
`@xlink:actuate`	OD default: `onRequest`	specifies the processing instructions, which can be either "`onLoad`" or "`onRequest`". This attribute shall not be present if the `@xlink:href` attribute is not present.
`@id`	O	specifies an identifier for this Period. The identifier shall be unique within the scope of the Media Presentation. If not present, no identifier for the Period is provided.
`AdaptationSet`	0...N	specifies an Adaptation Set. At least one Adaptation Set shall be present in each Period. However, the actual element may be present only in a remote element if xlink is in use, For more details see 5.3.3.

with:

Table 4 — Semantics of `Period` element

Element or Attribute Name	Use	Description
`Period`		specifies the information of a Period.
`@xlink:href`	O	specifies a reference to a remote element that contains one or multiple elements of type **`Period`**
`@xlink:actuate`	OD default: `onRequest`	specifies the processing instructions, which can be either "`onLoad`" or "`onRequest`". This attribute shall not be present if the `@xlink:href` attribute is not present.
`@id`	O	specifies an identifier for this Period. The idenfifier shall be unique within the scope of the Media Presentation. If the **`MPD`**`@type` is "`dynamic`", then this attribute shall be present and shall not change in case the MPD is updated. If not present, no identifier for the Period is provided.
`AdaptationSet`	0...N	specifies an Adaptation Set. At least one Adaptation Set shall be present in each Period unless the value of the `@duration` attribute of the Period is set to zero. Note that the actual element may be present only in a remote element if `xlink` is in use.For more details see 5.3.3.

5.3.3.1 Overview

Replace:

If there exist multiple media content components then the properties of each media content component shall be described by a separate **`ContentComponent`** element as defined in 5.5.4. The **`ContentComponent`** element shares common elements and attributes with the **`AdaptationSet`** element. Default values, or values applicable to all media content components, may be provided directly in the **`AdaptationSet`** element. Attributes present in the **`AdaptationSet`** shall not be repeated in the **`ContentComponent`** element.

with:

If there exist multiple media content components then the properties of each media content component shall be described by a separate **`ContentComponent`** element as defined in 5.3.4. The **`ContentComponent`** element shares common elements and attributes with the **`AdaptationSet`** element. Default values, or values applicable to all media content components, may be provided directly in the **`AdaptationSet`** element. Attributes present in the **`AdaptationSet`** shall not be repeated in the **`ContentComponent`** element.

5.3.3.2 Semantics

Replace:

Table 5 — Semantics of `AdaptationSet` element

Element or Attribute Name	Use	Description
`AdaptationSet`		Adaptation Set description
`@xlink:href`	O	specifies a reference to external **`AdaptationSet`** element
`@contentType`	O	specifies the media content component type for this Adaptation Set. A value of the top-level Content-type 'type' value as defined in RFC1521, Clause 4 shall be taken. If not present, the media content component type may be defined for each media component or it may be unknown.

with:

Table 5 — Semantics of `AdaptationSet` element

Element or Attribute Name	Use	Description
`AdaptationSet`		Adaptation Set description
`@xlink:href`	O	specifies a reference to a remote element that shall contain exactly one element of type **`AdaptationSet`**
`@contentType`	O	specifies the media content component type for this Adaptation Set. A value of the top-level Content-type 'type' value as defined in RFC4288, Clause 4 shall be taken. If not present, the media content component type may be defined for each media component or it may be unknown.

5.3.4.1 Overview

Replace:

The semantics of the attributes and elements within a **`ContentComponent`** element are provided in Table 6 of 5.3.3.2. The XML syntax of the **`ContentComponent`** element is provided in 5.3.3.3.

with:

The semantics of the attributes and elements within a **`ContentComponent`** element are provided in Table 6 of 5.3.4.2. The XML syntax of the **`ContentComponent`** element is provided in 5.3.4.3.

5.3.5.1 Overview

Replace:

For any dependent Representation X that depends on complementary Representation Y, the *m*-th Subsegment of X and the *n*-th Subsegment of Y shall be non-overlapping (as defined in 4.5.3) whenever *m* is not equal to *n*. For dependent Representations the concatenation of the Initialization Segment with the sequence of Subsegments of the dependent Representations, each being preceded by the corresponding Subsegment of each of the complementary Representations in order as provided in the `@dependencyId` attribute shall represent a conforming Subsegment sequence as defined in 4.5.3 conforming to the media format as specified in the `@mimeType` attribute for this dependent Representation.

NOTE When decoding of a dependent Representation is started from a SAP in the (Sub)Segment with number *i*, the decoding process does not need to access data from the complementary Representation(s) from any earlier (sub)segments than (sub)Segment with number *i* of the complementary Representation(s).

with:

For any dependent Representation X that depends on complementary Representation Y, the *m*-th Subsegment of X and the *n*-th Subsegment of Y shall be non-overlapping (as defined in 4.5.2) whenever *m* is not equal to *n*. For dependent Representations the concatenation of the Initialization Segment with the sequence of Subsegments of the dependent Representations, each being preceded by the corresponding Subsegment of each of the complementary Representations in order as provided in the `@dependencyId` attribute shall represent a conforming Subsegment sequence as defined in 4.5.3 conforming to the media format as specified in the `@mimeType` attribute for this dependent Representation.

NOTE When decoding of a dependent Representation is started from a SAP in the (Sub)Segment with number *i*, the decoding process does not need to access data from the complementary Representation(s) from any earlier (sub)segments than (sub)Segment with number *i* of the complementary Representation(s).

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

5.3.5.2 Semantics

Replace:

Table 7 —Semantics of `Representation` element

Element or Attribute Name	Use	Description
`Representation`		This element contains a description of a Representation.
`@id`	M	specifies an identifier for this Representation. The identifier shall be unique within a Period unless the Representation is functionally identically to another Representation in the same Period. The identifier shall not contain whitespace characters. If used in the template-based URL construction as defined in 5.3.9.4.4, the string shall only contain characters that are permitted within an HTTP-URL according to RFC 1738.
`@bandwidth`	M	Consider a hypothetical constant bitrate channel of bandwidth with the value of this attribute in bits per second (bps). Then, if the Representation is continuously delivered at this bitrate, starting at any SAP that is indicated either by `@startWithSAP` or by any Segment Index box, a client can be assured of having enough data for continuous playout providing playout begins after `@minBufferTime` * `@bandwidth` bits have been received (i.e. at time `@minBufferTime` after the first bit is received). For dependent Representations this value shall specify the minimum bandwidth as defined above of this Representation and all complementary Representations.

with:

Table 7 —Semantics of `Representation` element

Element or Attribute Name	Use	Description
`Representation`		This element contains a description of a Representation.
`@id`	M	specifies an identifier for this Representation. The identifier shall be unique within a Period unless the Representation is functionally identically to another Representation in the same Period. The identifier shall not contain whitespace characters. If used in the template-based URL construction as defined in 5.3.9.4.4, the string shall only contain characters that are permitted within an HTTP-URL according to RFC 3986.
`@bandwidth`	M	Consider a hypothetical constant bitrate channel of bandwidth with the value of this attribute in bits per second (bps). Then, if the Representation is continuously delivered at this bitrate, starting at any SAP that is indicated either by `@startWithSAP` or by any Segment Index box, a client can be assured of having enough data for continuous playout providing playout begins after `@minBufferTime` * `@bandwidth` bits have been received (i.e. at time `@minBufferTime` after the first bit is received). For dependent Representations this value specifies the bandwidth according to the above definition for the aggregation of this Representation and all complementary Representations.

5.3.7.2 Semantics

Replace:

`@codecs`	M	specifies the codecs present within the Representation. The codec parameters shall also include the profile and level information where applicable. The contents of this attribute shall conform to either the

		`simp-list` or `fancy-list` productions of RFC6381, Section 3.2, without the enclosing `DQUOTE` characters. The codec identifier for the Representation's media format, mapped into the name space for codecs as specified in RFC6381, Section 3.3, shall be used.
`ContentProtection`	0 … N	specifies information about content protection schemes used for the associated Representations. For details see 5.8.1 and 5.8.4.1.
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 @.		

with:

`@codecs`	O	specifies the codecs present within the Representation. The codec parameters shall also include the profile and level information where applicable. For segment formats defined in this specification this element shall be present and the contents of this attribute shall conform to either the `simp-list` or `fancy-list` productions of RFC6381, Section 3.2, without the enclosing `DQUOTE` characters. The codec identifier for the Representation's media format, mapped into the name space for codecs as specified in RFC6381, Section 3.3, shall be used.
`ContentProtection`	0 … N	specifies information about content protection schemes used for the associated Representations. For details see 5.8.1 and 5.8.4.1.
`EssentialProperty`	0 … N	specifies information about the containing element that is considered essential by the Media Presentation author for processing the containing element. For details see 5.8.4.8.
`SupplementalProperty`	0 … N	specifies supplemental information about the containing element that may be used by the DASH client optimizing the processing. For details see 5.8.4.9.
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 @.		

5.3.7.3 XML syntax

Replace:

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

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

  <!-- Video Scan type enumeration -->
  <xs:simpleType name="VideoScanType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="progressive"/>
      <xs:enumeration value="interlaced"/>
      <xs:enumeration value="unknown"/>
    </xs:restriction>
  </xs:simpleType>
```

with:

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

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

  <!-- Video Scan type enumeration -->
  <xs:simpleType name="VideoScanType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="progressive"/>
      <xs:enumeration value="interlaced"/>
      <xs:enumeration value="unknown"/>
    </xs:restriction>
  </xs:simpleType>
```

5.3.9.2.1 Overview

Replace:

In case multiple Media Segments are present, either a **`SegmentList`** or a **`SegmentTemplate`** is used that share the multiple Segment base information as provided in 5.3.9.2.3, Table 12.

with:

In case multiple Media Segments are present, either a **`SegmentList`** or a **`SegmentTemplate`** is used that share the multiple Segment base information as provided in 5.3.9.2.2, Table 12.

5.3.9.2.2 Semantics

Replace:

Table 11 — Semantics of `SegmentBase` element and *Segment Base Information* type

Element or Attribute Name	Use	Description
`SegmentBase` ***Segment Base Information***		specifies Segment base element as well as the type for the Segment base information.

with:

Table 11 — Semantics of `SegmentBase` element and *Segment Base Information* type

Element or Attribute Name	Use	Description
`SegmentBase` ***Segment Base Information***		specifies Segment base element. This element also specifies the type for the Segment base information that is the base type for other elements.

5.3.9.2.2 Semantics

Replace:

Segment Base Information		specifies Segment base information.

`@startNumber`	O	specifies the number of the first Media Segment in this Representation in the Period. For more details refer to 5.3.9.5.3.

with:

Segment Base Information		the Segment base information as specified in Table 11.

`@startNumber`	O	specifies the start number. The interpretation of the `@startNumber` depends on the segment addressing method. For more details refer to 5.3.9.5.3.

5.3.9.2.2 Semantics

Replace:

`@presentationTimeOffset`	O	specifies the presentation time offset of the Representation relative to the start of the Period. The value of the presentation time offset in seconds is the division of the value of this attribute and the value of the `@timescale` attribute. If not present on any level, the value of the presentation time offset is 0.
`@indexRange`	O	specifies the byte range that contains the Segment Index in all Media Segments of the

		Representation. The byte range shall be expressed and formatted as a `byte-range-spec` as defined in RFC 2616, Clause 14.35.1. It is restricted to a single expression identifying a contiguous range of bytes. If not present the value is unknown.

with:

`@presentationTimeOffset`	O	specifies the presentation time offset of the Representation relative to the start of the Period. The value of the presentation time offset in seconds is the division of the value of this attribute and the value of the `@timescale` attribute. If not present on any level, the value of the presentation time offset is 0.
`@timeShiftBufferDepth`	O	specifies the duration of the time shifting buffer for this Representation that is guaranteed to be available for a Media Presentation with type 'dynamic'. When not present, the value is of the `@timeShiftBufferDepth` on MPD level applies. If present, this value shall be not smaller than the value on MPD level. This value of the attribute is undefined if the type attribute is equal to 'static'. NOTE: When operating in a time-shift buffer on a Representation with value larger than the time-shift buffer than signalled on MPD level, not all Representations may be available for switching.
`@indexRange`	O	specifies the byte range that contains the Segment Index in all Media Segments of the Representation. The byte range shall be expressed and formatted as a `byte-range-spec` as defined in RFC 2616, Clause 14.35.1. It is restricted to a single expression identifying a contiguous range of bytes. If not present the value is unknown.

5.3.9.2.3 XML-Syntax

Replace:

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

  <!-- Multiple Segment information base -->
  <xs:complexType name="MultipleSegmentBaseType">
    <xs:complexContent>
      <xs:extension base="SegmentBaseType">
        <xs:sequence>
          <xs:element name="SegmentTimeline" type="SegmentTimelineType" minOccurs="0"/>
          <xs:element name="BitstreamSwitching" type="URLType" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="duration" type="xs:unsignedInt"/>
        <xs:attribute name="startNumber" type="xs:unsignedInt"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

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

with:

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

  <!-- Multiple Segment information base -->
  <xs:complexType name="MultipleSegmentBaseType">
    <xs:complexContent>
      <xs:extension base="SegmentBaseType">
        <xs:sequence>
          <xs:element name="SegmentTimeline" type="SegmentTimelineType" minOccurs="0"/>
          <xs:element name="BitstreamSwitching" type="URLType" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="duration" type="xs:unsignedInt"/>
        <xs:attribute name="startNumber" type="xs:unsignedInt"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

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

5.3.9.3.2 Semantics

Replace:

Table 14 — Semantics of `SegmentList` element

Element or Attribute Name	Use	Description
`SegmentList`		specifies Segment information.
`@xlink:href`	O	specifies a reference to external **`SegmentList`** element

`MultipleSegmentBaseInformation`		Multiple Segment base information as defined in 5.3.9.2.

with:

Table 14 — Semantics of `SegmentList` element

Element or Attribute Name	Use	Description
`SegmentList`		specifies Segment information.

Element or Attribute Name	Use	Description
@xlink:href	O	specifies a reference to a remote element that contains one or multiple elements of type **SegmentList**

MultipleSegmentBaseInformation		Multiple Segment base information as defined in 5.3.9.2, Table 12.

5.3.9.4.4 Template-based Segment URL construction

Replace:

Strings outside identifiers shall only contain characters that are permitted within URLs according to RFC 1738.

with:

Strings outside identifiers shall only contain characters that are permitted within URLs according to RFC 3986.

5.3.9.5.3 Media Segment information

Replace:

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

— A valid Media Segment URL and possibly a byte range,

— the number of the Media Segment in the Representation,

— the MPD start time of the Media Segment in the Representation providing an approximate presentation start time of the Segment,

— MPD duration of the Media Segment providing an approximate presentation duration of the Segment.

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

— if **SegmentTemplate** element is present the Template-based Segment URL construction in 5.3.9.4.4 shall be applied with the number of the Media Segment in the Media Segment list. The first number in the list is determined by the value of the **SegmentTemplate**@startNumber attribute, if present, or is 1 in case this attribute is not present.

— if one or more **SegmentList** elements are present they contain itself a list of **SegmentURL** elements for a consecutive list of Media Segment URLs. The first number in the list is determined by the value of the **SegmentList**@startNumber attribute, if present, or is 1 in case this attribute is not present. The sequence of multiple **SegmentList** elements within a Representation shall result in Media Segment List with consecutive numbers.

— none of the above: In this case only a single Media Segment shall be present with the URL provided by a **BaseURL** element and the **SegmentBase** element may be present.

The MPD start time is relative to the start of the Representation provided by the MPD. The MPD start time and the MPD duration are approximate and do not reflect the exact Media Presentation time. For more details on the relation of MPD start times and Media Presentation time refer to 7.2.1.

For the derivation of the MPD start time and duration of each Media Segment in the list of Media Segments, the *Number* of the Media Segment and the following information is used.

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

— If `@duration` attribute is present, then the MPD start time of the Media Segment is determined as ($Number$-$Number_{Start}$-1) times the value of the attribute `@duration` with $Number_{Start}$ the value of the `@startNumber` attribute. The MPD duration of the Media Segment is the value of the attribute `@duration` unless the Media Segment is the last one the Representation (see below for more details).

— If `@duration` attribute is not present and the **`SegmentTimeline`** element is present then rules in 5.3.9.6 apply to determine the start time and duration of each Media Segment in the Media Segment list.

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

For services with **`MPD`**`@type='dynamic'`, the Segment availability start time of a Media Segment is the sum of the value of the **`MPD`**`@availabilityStartTime`, the *PeriodStart* time of the containing Period as defined in 5.3.2.1, the MPD start time and the MPD duration of the Media Segment. The Segment availability end time of a Media Segment is the sum of the Segment availability start time, the MPD duration of the Media Segment and the value of the attribute **`MPD`**`@timeShiftBufferDepth`.

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

The data structures retrieved from the URL referring to a Media Segment are defined in 6.2.3.

with:

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

— A valid Media Segment URL and possibly a byte range,

— the number of the Media Segment in the Representation,

— the MPD start time of the Media Segment in the Representation providing an approximate presentation start time of the Segment,

— MPD duration of the Media Segment providing an approximate presentation duration of the Segment.

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

— if **`SegmentTemplate`** element is present the Template-based Segment URL construction in 5.3.9.4.4 shall be applied with the number of the Media Segment in the Media Segment list. If the Representation contains or inherits a **`SegmentTemplate`** element with *$Number$* then the URL of the media segment at position *k* is determined by replacing the $*Number*$ identifier by (*k*-1) + `@startNumber`. If the Representation contains or inherits a **`SegmentTemplate`** element with *$Time$* then the URL of the media segment at position *k* is determined by replacing the $*Time*$

identifier by the time address associated to this segment. The time address is determined as follows:

— if the `@duration` attribute is present, then the time address is determined by replacing the $*Time*$ identifier with ((*k*-1) + `@startNumber)* @duration.` Further, if the Segment does not contain the Segment Index, then the media time of the Segment shall be accurately expressed by the MPD information in the following sense:

  — the value ((*k*-1) + `@startNumber`)* `@duration` shall be identical to the earliest presentation time in the segment.
  — the duration of the segment in media presentation time shall be identical to the value of the `@duration` attribute.

— if the **`SegmentTimeline`** element is present, then the time address is determined by replacing the $*Time*$ identifier with the earliest presentation time of the k-th segment as documented in the Segment timeline in 5.3.9.6.

— if one or more **`SegmentList`** elements are present they contain itself a list of **`SegmentURL`** elements for a consecutive list of Media Segment URLs. The first number in the list within this Period is determined by the value of the **`SegmentList`**`@startNumber` attribute, if present, or is 1 in case this attribute is not present. The sequence of multiple **`SegmentList`** elements within a Representation shall result in Media Segment List with consecutive numbers.

— none of the above: In this case only a single Media Segment shall be present with the URL provided by a **`BaseURL`** element and the **`SegmentBase`** element may be present.

The MPD start time is relative to the start of the Representation provided by the MPD. The MPD start time and the MPD duration are approximate and do not reflect the exact Media Presentation time. For more details on the relation of MPD start times and Media Presentation time refer to 7.2.1.
For the derivation of the MPD start time and duration of each Media Segment in the list of Media Segments, the *Number* of the Media Segment and the following information is used.

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

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

— If `@duration` attribute is not present and the **`SegmentTimeline`** element is present then rules in 5.3.9.6 apply to determine the start time and duration of each Media Segment in the Media Segment list.

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

For services with **`MPD`**`@type='dynamic'`, the Segment availability start time of a Media Segment is the sum of the value of the **`MPD`**`@availabilityStartTime`, the *PeriodStart* time of the containing Period as defined in 5.3.2.1, the MPD start time and the MPD duration of the Media Segment. The Segment availability end time of a Media Segment is the sum of the Segment availability start time, the MPD duration of the Media Segment and the value of the attribute `@timeShiftBufferDepth` for this Representation.

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

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

The data structures retrieved from the URL referring to a Media Segment are defined in 6.2.3.

5.3.9.6.1 General

Replace:

The **`SegmentTimeline`** element shall contain a list of **`S`** elements each of which describes a sequence of contiguous segments of identical MPD duration. The **`S`** element contains a mandatory `@d` attribute specifying the MPD duration, an optional `@r` repeat count attribute specifying the number of contiguous Segments with identical MPD duration minus one and an an optional `@t` time attribute specifying the MPD start time of the first Segment in the series.

The `@r` attribute has a default value of zero (i.e., a single Segment in the series) when not present. For example, a repeat count of three means there are four contiguous Segments, each with the same MPD duration.

with:

The **`SegmentTimeline`** element shall contain a list of **`S`** elements each of which describes a sequence of contiguous segments of identical MPD duration. The **`S`** element contains a mandatory `@d` attribute specifying the MPD duration, an optional `@r` repeat count attribute specifying the number of contiguous Segments with identical MPD duration minus one and an optional `@t` time attribute. The value of the `@t` attribute minus the value of the `@presentationTimeOffset` specifies the MPD start time of the first Segment in the series.

The `@r` attribute has a default value of zero (i.e., a single Segment in the series) when not present. For example, a repeat count of three means there are four contiguous Segments, each with the same MPD duration. The value of the `@r` attribute of the **`S`** element may be set to a negative value indicating that the duration indicated in `@d` is promised to repeat until the **`S`**`@t` of the next **`S`** element or if it is the last **`S`** element in the **`SegmentTimeline`** element until the end of the Period or the next update of the MPD, i.e. it is treated in the same way as the `@duration` attribute for a full period.

5.3.9.6.2 Semantics

Replace:

`@r`	OD default: 0	specifies the repeat count of the number of following contiguous Segments with the same duration expressed by the value of `@d`. This value is zero-based (e.g. a value of three means four Segments in the contiguous series).

with:

`@r`	OD default: 0	specifies the repeat count of the number of following contiguous Segments with the same duration expressed by the value of `@d`. This value is zero-based (e.g. a value of three means four Segments in the contiguous series). A negative value of the `@r` attribute of the **S** element indicates that the duration indicated in `@d` attribute repeats until the start of the next **S** element, the end of the Period or until the next MPD update.

5.3.9.6.3 XML syntax

Replace:

```
  <!-- Segment Timeline -->
  <xs:complexType name="SegmentTimelineType">
    <xs:sequence>
      <xs:element name="S" minOccurs="1" maxOccurs="unbounded" >
        <xs:complexType>
          <xs:attribute name="t" type="xs:unsignedInt"/>
          <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:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
```

with:

```
  <!-- Segment Timeline -->
  <xs:complexType name="SegmentTimelineType">
    <xs:sequence>
      <xs:element name="S" minOccurs="1" maxOccurs="unbounded" >
        <xs:complexType>
          <xs:attribute name="t" type="xs:unsignedLong"/>
          <xs:attribute name="d" type="xs:unsignedLong" use="required"/>
          <xs:attribute name="r" type="xs:int" use="optional" default="0"/>
          <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:complexType>
      </xs:element>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
```

5.4 Media Presentation Description updates

Replace:

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

with:

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

5.5.3 Processing

Replace:

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

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

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

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

4) Any URI reference to a remote element shall be an HTTP-URL.

5) If a URI reference is relative then reference resolution as defined in 5.6.4 shall apply.

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

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

2) The remote 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 element.

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

with:

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

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

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

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

4) Any URI reference to a remote element shall be an HTTP-URL.

5) If a URI reference is relative then reference resolution as defined in 5.6.4 shall apply.

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

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

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

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

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

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

5.8.1 General

Replace:

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

In this part of ISO/IEC 23009, specific elements for descriptors are defined in 5.8.4.

The MPD does not provide any specific information on how to use these elements. It is up to the application that employs DASH formats to instantiate the description elements with appropriate scheme information. However, this part of ISO/IEC 23009 defines some specific schemes in 5.8.5.

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

In the case that a simple set of enumerated values are required, a text string may be defined for each value and this string must be included in the `@value` attribute. If structured data is required then any extension element or attribute may be defined in a separate namespace.Two elements of type `DescriptorType` are *equivalent*, if the element name, the value of the `@schemeIdUri` and the value of the `@value` attribute are equivalent. If the `@schemeIdUri` is a URN, then equivalence shall refer to lexical equivalence as defined in clause 5 of RFC 2141. If the `@schemeIdUri` is a URL, then equivalence shall refer to equality on a character-for-character basis as defined in clause 6.2.1 of RFC3986. If the `@value` attribute is not present, equivalence is determined by the equivalence for `@schemeIdUri` only. Attributes and element in extension namespaces are not used for determining equivalence.

with:

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

In this part of ISO/IEC 23009, specific elements for descriptors are defined in 5.8.4.

The MPD does not provide any specific information on how to use these elements. It is up to the application that employs DASH formats to instantiate the description elements with appropriate scheme information. However, this part of ISO/IEC 23009 defines some specific schemes in 5.8.5.

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

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

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

Two elements of type DescriptorType are *equivalent*, if the element name, the value of the @schemeIdUri and the value of the @value attribute are equivalent. If the @schemeIdUri is a URN, then equivalence shall refer to lexical equivalence as defined in clause 5 of RFC 2141. If the @schemeIdUri is a URL, then equivalence shall refer to equality on a character-for-character basis as defined in clause 6.2.1 of RFC3986. If the @value attribute is not present, equivalence is determined by the equivalence for @schemeIdUri only. Attributes and element in extension namespaces are not used for determining equivalence. The @id attribute may be ignored for equivalence determination.

5.8.2 Semantics of generic descriptor

Replace:

@value	O	specifies the value for the descriptor element. The value space and semantics must be defined by the owners of the scheme identified in the @schemeIdUri attribute.

with:

@value	O	specifies the value for the descriptor element. The value space and semantics must be defined by the owners of the scheme identified in the @schemeIdUri attribute.
@id	O	specifies an identifier for the descriptor. Descriptors with identical values for this attribute shall be synonymous, i.e. the processing of one of the descriptors with an identical value is sufficient.

5.8.3 XML syntax of generic descriptor

Replace:

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

with:

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

5.8.4.6 Frame-packing

Replace:

This part of ISO/IEC 23009 defines frame-packing schemes in 5.8.5.6.

with:

The descriptor may carry frame-packing schemes using the URN label and values defined for `VideoFramePackingType` in ISO/IEC 23001-8.

NOTE: This part of ISO/IEC 23009 also defines frame-packing schemes in 5.8.5.6. These schemes are maintained for backward-compatibility, but it is recommended to use the signalling as defined in ISO/IEC 23001-8.

5.8.4.7 Audio channel configuration

Replace:

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

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

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

A scheme for audio channel configuration is defined in 5.8.5.4 of this part of ISO/IEC 23009.

with:

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

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

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

The descriptor may carry audio channel configuration using the URN label and values defined for `OutputChannelPosition` in ISO/IEC 23001-8.

NOTE A scheme for audio channel configuration is also defined in 5.8.5.4 of this part of ISO/IEC 23009. This scheme is maintained for backward-compatibility, but it is recommended to use the signalling as defined in ISO/IEC 23001-8.

Add the following sections after 5.8.4.7:

5.8.4.8 Essential Property Descriptor

For the element **`EssentialProperty`** the Media Presentation author expresses that the successful processing of the descriptor is essential to properly use the information in the parent element that contains this descriptor unless the element shares the same `@id` with another **`EssentialProperty`** element.
If **`EssentialProperty`** elements share the same `@id`, then processing one of the **`EssentialProperty`** elements with the same value for `@id` is sufficient. At least one **`EssentialProperty`** element of each distinct `@id` value is expected to be processed.

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

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

5.8.4.9 Supplemental Property Descriptor

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

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

Multiple **SupplementalProperty** elements may be present.

5.8.5.2 Content protection

Replace:

In this scheme, the value of the `@value` attribute shall be the 4CC contained in the Scheme Type Box, suitably escaped according to RFC 2141.

with:

In this scheme, the value of the `@value` attribute shall be the 4CC contained in the Scheme Type Box, suitably escaped according to RFC 2141 and may include the version number. The 4CC and the version number, if present, shall be separated by a ":". The version number shall be encoded as up to 8 hexadecimal digits, where the leading ‘0’s may be omitted.

5.8.5.3 Frame-packing

Replace:

The following defines a set of URIs that identify specific frame-packing arrangments, i.e. schemes contained in the **FramePacking** element:

— For Adaptation Sets or Representations that contain a video component that conforms to ISO/IEC 14496-10, the URI `urn:mpeg:dash:14496:10:frame_packing_arrangement_type:2011` is defined to indicate the frame-packing arrangement 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 Adaptation Sets or Representations that contain a video component that conforms to ISO/IEC 13818-1, the URI `urn:mpeg:dash:13818:1:stereo_video_format_type:2011` is defined to indicate the frame-packing arrangement as defined by Table L-1 of ISO/IEC 13818-1 (‘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.

with:

The following defines a set of URIs that identify specific frame-packing arrangements, i.e. schemes contained in the **FramePacking** element:

— For Adaptation Sets or Representations that contain a video component that conforms to ISO/IEC 14496-10, the URI `urn:mpeg:dash:14496:10:frame_packing_arrangement_type:2011` is defined. The `@value` shall be value as defined for `VideoFramePackingType` in ISO/IEC 23001-8.

— For Adaptation Sets or Representations that contain a video component that conforms to ISO/IEC 13818-1, the URI `urn:mpeg:dash:13818:1:stereo_video_format_type:2011` is defined. The `@value` shall be the values as defined for `VideoFramePackingType` in ISO/IEC 23001-8.

5.8.5.4 Audio channel configuration scheme

Replace:

The following defines a URI that identifies channel configuration signalling for Representations that contain an audio component. The URI "`urn:mpeg:dash:23003:3:audio_channel_configuration:2011`" is defined to indicate the channel configuration as defined by Table 68 (Channel Configurations, meaning of `channelConfigurationIndex`, mapping of channel elements to loudspeaker positions') of ISO/IEC 23003-3. The `@value` shall be the 'value' column as specified in Table 68 and shall be interpreted according to the remaining columns in the same table.

with:

The following defines a URI that identifies channel configuration signalling for Representations that contain an audio component. The URI "`urn:mpeg:dash:23003:3:audio_channel_configuration:2011`" is defined to indicate the channel configuration. The `@value` shall be the value as defined for `OutputChannelPosition` in ISO/IEC 23001-8.

6.3.5.1 General format type

Replace:

The Self-Initializing Media Segment is conformant with the ISO base media file format.

with:

The Self-Initializing Media Segment is conformant with the ISO base media file format and defines the DASH Self-Initializing Media Segment '`dsms`' brand.

6.4.3.2 Initialization Segment

Replace:

Initialization Segment may or may not be present. If it is not present for a given Representation, all Media Segments belonging to this Representation shall be self-initializing. Also, if an Initialization Segment is used, not all initialization information needs to reside in the Initialization Segment, only presence of complete initialization information in the concatenation of Initialization Segment and Media Segment is required.

with:

The Initialization Segment shall contain only complete sections.

The Initialization Segment may or may not be present. If it is not present for a given Representation, all Media Segments belonging to this Representation shall be self-initializing. Also, if an Initialization Segment is used, not all initialization information needs to reside in the Initialization Segment, only presence of complete initialization information in the concatenation of Initialization Segment and Media Segment is required.

6.4.4.2 Basic Media Segment

Replace:

Media Segments should contain only complete PES packets. Each PES packet should be comprised of one or more complete access units in each packet. Media Segments should contain only complete access units.

with:

Media Segments should contain only complete PES packets and sections. Each PES packet should be comprised of one or more complete access units in each packet. Media Segments should contain only complete access units.

7.4.3.2 Segment alignment

Replace:

If the `@segmentAlignment` attribute is not set to '`false`', the requirements stated in 5.3.2 and 5.3.3.2 shall be met. In addition, the Media Segment shall contain only complete PES packets and only complete access units for each PID, and the first PES packet shall contain a PTS timestamp.

with:

If the `@segmentAlignment` attribute is not set to '`false`', the requirements stated in 5.3.2 and 5.3.3.2 shall be met. In addition, the Media Segment shall contain only complete PES packets and sections and only complete access units for each PID, and the first PES packet shall contain a PTS timestamp.

7.4.3.3 Subsequent alignment

Replace:

If the `@subsegmentAlignment` flag is not set to '`false`', the semantics as defined in 5.3.3.2 shall apply. In particular, for an MPEG-2 TS-based Media Presentation, a Subsegment shall contain only complete PES packets for each PID, and the first PES packet from each elementary stream shall contain a PTS.

with:

If the `@subsegmentAlignment` flag is not set to '`false`', the semantics as defined in 5.3.3.2 shall apply. In particular, for an MPEG-2 TS-based Media Presentation, a Subsegment shall contain only complete PES packets and sections for each PID, and the first PES packet from each elementary stream shall contain a PTS.

8.1. Definition

Add the following paragraph at the end of 8.1:

External organizations or individuals may define restrictions, permissions and extensions by using this profile mechanism. It is recommended that such external definitions be not referred to as profiles, but as *Interoperability Points*. Such an interoperability points may be signalled in the `@profiles` parameter once a URI is defined. The owner of the URI is responsible to provide sufficient semantics on the restrictions and permission of this interoperability point.

8.4.2 Media Presentation Description constraints

Add the last bullet point to the current list:

The Media Presentation Description shall conform to the following constraints:

— The rules for the MPD and segments as defined in 7.3 shall apply.

— Representations not inferred to have `@profiles` equal to the profile identifier as defined in 8.4.1 may be ignored.

— In addition, **`Representation`** elements contained in an **`AdaptationSet`** element complying to this profile shall have the following constraints:

    — **`Representation`** elements with `@startWithSAP` value (either supplied directly or inherited from the containing **`AdaptationSet`**) equal to 3 may be ignored if both the following conditions hold:
        — the containing Adaptation Set contains more than one Representation, and
        — no other Representation has the same value for `@mediaStreamStructureId`.
    — The **`SegmentTemplate`** element shall be present on at least one of the three levels, the Period level containing the Representation, the Adaptation Set containing the Representation, or on Representation level itself.
    — **`Representation`** elements with a `@startWithSAP` value (either supplied directly or inherited from the containin) absent, zero or greater than 3 may be ignored.
    — **`AdaptationSet`** elements with a `@segmentAlignment` value '`false`' or absent may be ignored.
    — **`Representation`** elements with `@startWithSAP` value (either supplied directly or inherited from the containing Adaptation Set) equal to 3 may be ignored if both of the following conditions hold:
        — the containing Adaptation Set contains more than one Representation, and
        — no other Representation has the same value for `@mediaStreamStructureId`.

— **`Subset`** elements may be ignored.

— Elements using the `@xlink:href` attribute may be ignored from the MPD. The Representations conforming to this profile are those not accessed through an Adaptation Set that uses an `@xlink:href`.

— When the MPD is updated, the value of **`MPD`**`@availabilityStartTime` shall be the same in the original and the updated MPD

8.6.2 Media Presentation Description constraints

Add the last bullet point to the current list:

The Media Presentation Description shall conform to the following constraints:

— The rules for the MPD as defined in 7.4 shall apply.

— Representations not complying with the restrictions defined in 7.4 or not inferred to have `@profiles` equal to the profile identifier as defined in 8.6.1 may be ignored.

— Representations not in group 0 may be ignored;

— **`Subset`** may be ignored;

— Representations containing the **`SegmentTimeline`** element may be ignored;

— It shall be possible to present a presentation conforming to this profile without resolving `@xlink:href` in **`AdaptationSet`** or **`SegmentList`** elements. Any initial **`Period`** elements using `@xlink:href` may be ignored, and the first non-excluded Period must have an explicit `@start` attribute. After the first non-excluded Period, there shall be no Period using `@xlink:href`.

— When the MPD is updated, the value of **MPD**@availabilityStartTime shall be the same in the original and the updated MPD

8.7.2 Media Presentation Description constraints

Add the last bullet point to the current list:

The Media Presentation Description shall conform to the following constraints:

— All MPD constraints of MPEG-2 TS Main Profile as defined in 8.6.2 shall be obeyed;

— Representations not complying with the restrictions defined in 7.4 or not inferred to have @profiles equal to the profile identifier as defined in 8.7.1 may be ignored.

— If an Index Segment is provided, any Adaptation Set with @subsegmentAlignment set to 'false' may be ignored;

— Any Adaptation Set, which contains more than one Representation and has @bitstreamSwitching not set to 'true', may be ignored;

— When the MPD is updated, the value of **MPD**@availabilityStartTime shall be the same in the original and the updated MPD

A.2 Overview

Replace:

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

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

6) With the wall-clock time *NOW* advancing, the client consumes the available Segments. As *NOW* advances the client possibly expands the list of available Segments for each Representation according to the procedures specified in A.3 If the following conditions 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

with:

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

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

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

   i) if the attribute `MPD@minimumUpdatePeriod` is present and

A.3 Segment list generation

Replace:

A.3.1 General

Assume that the DASH client has access to an MPD. This clause describes how a client may generate a Segment list for one Representation as shown in Table A.1 from an MPD obtained at *FetchTime* at a specific client-local time *NOW*. In this description, the term *NOW* is used to refer to "the current value of the clock at the reference client when performing the construction of an MPD Instance from an MPD". A client that is not synchronized with a DASH server, which is in turn is expected to be synchronized to UTC, may experience issues in accessing Segments as the Segment availability times provided by the server and the local time *NOW* may not be synchronized. Therefore, DASH clients are expected to synchronize their clocks to a globally accurate time standard.

Table A.1 — Segment list in example client

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

According to 5.3.9 there exist three different ways to describe and generate a Segment List. This description focuses on the first two where either a **`SegmentList`** element or a **`SegmentTemplate`** element is present.

The case with a single Media Segment using **`BaseURL`** element and **`SegmentBase`** element is considered straightforward.

a) If the Representation contains or inherits a **`SegmentTemplate`** element, then the procedures in A.3.2 are used to generate a list of Media Segments.

b) If the Representation contains or inherits one or more **`SegmentList`** elements, providing a set of explicit URL(s) for Media Segments, then the procedures in A.3.3 are used to generate a list of Media Segments.

c) If the **`MPD`**`@type` attribute is `'dynamic'`, then the restrictions on Media Segment Lists as provided in A.3.4 need to be taken into account.

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

A.3.2 Template-based generation of Segment list

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

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

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

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

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

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

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

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

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

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

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

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

6) A new Media Segment is added to the list, i.e. *i* = *i* + 1, MediaSegment.StartTime[*i*] = MediaSegment.StartTime[*i*-1] + `@duration` and the guaranteed duration is set to MediaSegment.duration[*i*] = *PeriodEndTime* - MediaSegment.StartTime[*i*] The restrictions as specified in A.3.4 are applied for the creation of the accessible list of Media Segments and this concludes Segment List generation.

If instead of the `@duration` attribute a **`SegmentTimeline`** element is given, then the variable durations of the Segments are used to compute the start times and durations. Also, depending on the identifier, *$Number$* or *$Time$,* the appropriate replacements are done as introduced in 5.3.9.6. If neither the `@duration` nor the **`SegmentTimeline`** element is given, then the *MediaSegment.StartTime*[*1*] of the only provided Segment is set to 0.

A.3.3 Playlist-based generation of Segment list

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

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

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

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

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

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

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

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

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

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

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

A.3.4 Media Segment list restrictions

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

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

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

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

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

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

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

with:

A.3.1 General

Assume that the DASH client has access to an MPD. This clause describes how a client may generate a Segment list for one Representation as shown in Table A.1 from an MPD obtained at *FetchTime* at a specific client-local time *NOW*. In this description, the term *NOW* is used to refer to "the current value of the clock at the reference client when performing the construction of an MPD Instance from an MPD". A client that is not synchronized with a DASH server, which is in turn is expected to be synchronized to UTC, may experience issues in accessing Segments as the Segment availability times provided by the server and the local time *NOW* may not be synchronized. Therefore, DASH clients are expected to synchronize their clocks to a globally accurate time standard.

Table A.1 — Segment list in example client

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

According to 5.3.9 there exist three different ways to describe and generate a Segment List. This description focuses on the first two where either a `SegmentList` element or a `SegmentTemplate` element is present.

The case with a single Media Segment using **`BaseURL`** element and **`SegmentBase`** element is considered straightforward.

Segments are available at its assigned URL if at wall-clock time *NOW* the Segment availability start time is smaller than or equal to *NOW* and the Segment availability end time is larger than or equal to *NOW*. Furthermore, assume that for a Representation in a Period, the Segment list is indexed with *i*=1, ..., *N*.

A.3.2 Period Start and End Times

Assume that for an MPD with fetch time *FetchTime*

— the Period start time is provided as *PeriodStart* according to 5.3.2.1 for any Period in the MPD.

— the Period end time referred as *PeriodEnd* is determined as follows: For any Period in the MPD except for the last one, the *PeriodEnd* is obtained as the value of the *PeriodStart* of the next Period. For the last Period in the MPD:

  — if the **`MPD`**`@minimumUpdatePeriod` attribute is not present, then *PeriodEnd* is defined as the end time of the Media Presentation, i.e. **`MPD`**`@availabilityStartTime` + **`MPD`**`@mediaPresentationDuration`.

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

A.3.3 Start Time and Duration

In case the Segment base information contains the `@duration` attribute, then

— the regular duration *d* is obtained as *d* = `@duration/@timescale`,

— the MPD start time MediaSegment[*i*].startTime is obtained as (*i-1*)**d*,

— the MPD duration MediaSegment[*i*].duration is obtained as *d* unless this Segment is the last Segment in this Period, then the MediaSegment[*i*].duration is obtained as *PeriodEnd* - MediaSegment.StartTime[*i*]).

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

— the *t*[*s*] is the value of `@t` of the *s*-th **`S`** element divided by the value of the `@timescale` attribute,

— the *o* is the value of `@presentationTimeOffset` for this Representation divided by the value of the `@timescale` attribute,

— the *d*[*s*] is the value of `@d` of the *s*-th **`S`** element divided by the value of the `@timescale` attribute,

— if the value of `@r` is greater equal than 0

  — the *r*[*s*] is one more than the value of `@r` of the *s*-th **`S`** element and

  — *N*=0

  — for *s=1, ...* N_S

    — *N* = *N* + 1

    — MediaSegment[*N*].startTime = *t*[*s*] - *o*
    — MediaSegment[*N*].duration = *d*[*s*]
    — for *j* = 1, ..., *r*[*s*]
        — *N* = *N* + 1
        — MediaSegment[*N*].startTime = MediaSegment[*N-1*].startTime + *d*[*s*]
        — MediaSegment[*N*].duration = *d*[*s*]

- else
  - the MPD duration MediaSegment[*i*].duration is obtained as *d*[0] unless this Segment is the last Segment in this Period, then the MediaSegment[*i*].duration is obtained as *PeriodEnd* - MediaSegment.StartTime[*i*]).

If neither the `@duration` nor the **`SegmentTimeline`** element is given, then

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

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

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

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

A.3.4 Media Segment list restrictions

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

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

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

In case of MPD updates, assume the variable *CheckTime* associated to an MPD with *FetchTime* is defined as the sum of the fetch time of this operating MPD and the value of the attribute **`MPD`**`@minimumUpdatePeriod`, i.e. *CheckTime* = *FetchTime* + **`MPD`**`@minimumUpdatePeriod`. The *CheckTime* is defined on the MPD-documented media time axis; when the client's playback time reaches *CheckTime* - **MPD**@minBufferTime it should fetch a new MPD.

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

A.4, Seeking

Replace:

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

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

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

with:

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

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

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

Annex B, MPD Schema

Replace:

```
<?xml version="1.0"?>
<xs:schema targetNamespace="urn:mpeg:DASH:schema:MPD:2011"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns="urn:mpeg:DASH:schema:MPD:2011">

  <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 the Media Presentation Description for MPEG-DASH.
    </xs:documentation>
  </xs:annotation>

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

  <!-- MPD Type -->
  <xs:complexType name="MPDtype">
    <xs:sequence>
      <xs:element name="ProgramInformation" type="ProgramInformationType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Location" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Period" type="PeriodType" maxOccurs="unbounded"/>
      <xs:element name="Metrics" type="MetricsType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string"/>
    <xs:attribute name="profiles" type="xs:string" use="required"/>
    <xs:attribute name="type" type="PresentationType" default="static"/>
    <xs:attribute name="availabilityStartTime" type="xs:dateTime"/>
    <xs:attribute name="availabilityEndTime" type="xs:dateTime"/>
    <xs:attribute name="mediaPresentationDuration" type="xs:duration"/>
    <xs:attribute name="minimumUpdatePeriod" type="xs:duration"/>
    <xs:attribute name="minBufferTime" type="xs:duration" use="required"/>
    <xs:attribute name="timeShiftBufferDepth" type="xs:duration"/>
    <xs:attribute name="suggestedPresentationDelay" type="xs:duration"/>
    <xs:attribute name="maxSegmentDuration" type="xs:duration"/>
    <xs:attribute name="maxSubsegmentDuration" type="xs:duration"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

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

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

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

  <!-- Ratio Type for sar and par -->
  <xs:simpleType name="RatioType">
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9]*:[0-9]*"/>
    </xs:restriction>
  </xs:simpleType>

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


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

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

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

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


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

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

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

  <!-- Video Scan type enumeration -->
  <xs:simpleType name="VideoScanType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="progressive"/>
      <xs:enumeration value="interlaced"/>
      <xs:enumeration value="unknown"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- Subset  -->
  <xs:complexType name="SubsetType">
    <xs:attribute name="contains" type="UIntVectorType" use="required"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
```

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

<!-- Multiple Segment information base -->
<xs:complexType name="MultipleSegmentBaseType">
  <xs:complexContent>
    <xs:extension base="SegmentBaseType">
      <xs:sequence>
        <xs:element name="SegmentTimeline" type="SegmentTimelineType" minOccurs="0"/>
        <xs:element name="BitstreamSwitching" type="URLType" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="duration" type="xs:unsignedInt"/>
      <xs:attribute name="startNumber" type="xs:unsignedInt"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

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

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

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

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

<!-- Segment Timeline -->
<xs:complexType name="SegmentTimelineType">
  <xs:sequence>
    <xs:element name="S" minOccurs="1" maxOccurs="unbounded" >
      <xs:complexType>
```

```
          <xs:attribute name="t" type="xs:unsignedInt"/>
          <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:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

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

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

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

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

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

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

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

</xs:schema>
```

with:

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

```
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns="urn:mpeg:dash:schema:mpd:2011">

   <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 the Media Presentation Description for MPEG-DASH.
     </xs:documentation>
   </xs:annotation>

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

   <!-- MPD Type -->
   <xs:complexType name="MPDtype">
     <xs:sequence>
       <xs:element name="ProgramInformation" type="ProgramInformationType" minOccurs="0"
 maxOccurs="unbounded"/>
       <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
       <xs:element name="Location" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
       <xs:element name="Period" type="PeriodType" maxOccurs="unbounded"/>
       <xs:element name="Metrics" type="MetricsType" minOccurs="0" maxOccurs="unbounded"/>
       <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:attribute name="id" type="xs:string"/>
     <xs:attribute name="profiles" type="xs:string" use="required"/>
     <xs:attribute name="type" type="PresentationType" default="static"/>
     <xs:attribute name="availabilityStartTime" type="xs:dateTime"/>
     <xs:attribute name="availabilityEndTime" type="xs:dateTime"/>
     <xs:attribute name="mediaPresentationDuration" type="xs:duration"/>
     <xs:attribute name="minimumUpdatePeriod" type="xs:duration"/>
     <xs:attribute name="minBufferTime" type="xs:duration" use="required"/>
     <xs:attribute name="timeShiftBufferDepth" type="xs:duration"/>
     <xs:attribute name="suggestedPresentationDelay" type="xs:duration"/>
     <xs:attribute name="maxSegmentDuration" type="xs:duration"/>
     <xs:attribute name="maxSubsegmentDuration" type="xs:duration"/>
     <xs:anyAttribute namespace="##other" processContents="lax"/>
   </xs:complexType>

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

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

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

   <!-- Ratio Type for sar and par -->
   <xs:simpleType name="RatioType">
     <xs:restriction base="xs:string">
       <xs:pattern value="[0-9]*:[0-9]*"/>
     </xs:restriction>
   </xs:simpleType>

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


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

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

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

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


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

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

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

   <!-- Video Scan type enumeration -->
   <xs:simpleType name="VideoScanType">
     <xs:restriction base="xs:string">
       <xs:enumeration value="progressive"/>
       <xs:enumeration value="interlaced"/>
       <xs:enumeration value="unknown"/>
     </xs:restriction>
   </xs:simpleType>

   <!-- Subset  -->
   <xs:complexType name="SubsetType">
     <xs:attribute name="contains" type="UIntVectorType" use="required"/>
     <xs:anyAttribute namespace="##other" processContents="lax"/>
   </xs:complexType>

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

   <!-- Multiple Segment information base -->
   <xs:complexType name="MultipleSegmentBaseType">
     <xs:complexContent>
       <xs:extension base="SegmentBaseType">
         <xs:sequence>
           <xs:element name="SegmentTimeline" type="SegmentTimelineType" minOccurs="0"/>
           <xs:element name="BitstreamSwitching" type="URLType" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="duration" type="xs:unsignedInt"/>
         <xs:attribute name="startNumber" type="xs:unsignedInt"/>
       </xs:extension>
     </xs:complexContent>
   </xs:complexType>

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

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

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

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

   <!-- Segment Timeline -->
   <xs:complexType name="SegmentTimelineType">
     <xs:sequence>
       <xs:element name="S" minOccurs="1" maxOccurs="unbounded" >
         <xs:complexType>
           <xs:attribute name="t" type="xs:unsignedLong"/>
           <xs:attribute name="d" type="xs:unsignedLong" use="required"/>
           <xs:attribute name="r" type="xs:integer" use="optional" default="0"/>
           <xs:anyAttribute namespace="##other" processContents="lax"/>
         </xs:complexType>
       </xs:element>
       <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:anyAttribute namespace="##other" processContents="lax"/>
   </xs:complexType>

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

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

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

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

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

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

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

 </xs:schema>
```

C.2 MIME type and subtype

Replace:

C.2 MIME type and subtype

The MIME Type and Subtype are defined as follows:

— MIME media type name:	application
— MIME subtype name:	dash+xml
— Required parameters:	none
— Optional parameters:	The '`profiles`' parameter as documented in Annex C.3.1 in this part of ISO/IEC 23009
— Encoding considerations:	the same media type encoding considerations specified in 3.2 of RFC 3023
— 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 23009-1: Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats

— Applications which use this media type:	various
— Additional information:	
— File extension(s):	mpd
— Intended usage:	common
— Other Information/General Comment:	none
— Person to contact for further information:	
— Name:	Thomas Stockhammer
— Email:	stockhammer@nomor.de
— Author/Change controller:	ISO/IEC JTC1/SC29 (MPEG)

with:

The MIME Type and Subtype are defined as follows:

— MIME media type name:	application
— MIME subtype name:	dash+xml
— Required parameters:	none
— Optional parameters:	The '`profiles`' parameter as documented in Annex C.3.1 in this part of ISO/IEC 23009
— Encoding considerations:	UTF-8
— 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. The Media Presentation Description (MPD) format does not incorporate any active or executable content. However, other forms of material from outside sources can be referenced by an MPD, and this material could contain active or executable content. Such material is expected to be identified by its own MIME type, and the security considerations of that format should be taken into account. If operating in an insecure environment and required by the content/service provider, elements and attributes of MPD may be encrypted to protect their confidentiality by using the syntax and processing rules specified in the W3C Recommendation "XML Encryption Syntax and Processing".