INTERNATIONAL STANDARD

ISO/IEC 23009-2

Third edition
2020-09

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

Part 2: Conformance and reference software

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

Partie 2: Conformité et logiciel de référence

Reference number
ISO/IEC 23009-2:2020(E)

Contents

Foreword

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

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

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

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

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see: www.iso.org/iso/foreword.html.

This document was prepared by Joint Techncial Committee ISO/IEC JTC 1, *Information technology*, SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This third edition cancels and replaces the second edition (ISO/IEC 23009-2:2017), which has been technically revised.

The main changes compared to the previous edition are as follows:

a) Test vectors, conformance and reference software to cover all the features of ISO/IEC 23009-5, including:

   — SAND HTTP conformance client;

   — SAND HTTP conformance server;

   — SAND WebSocket conformance server.

b) Test vecors, conformance and reference software to cover all the features of ISO/IEC 23009-6, including:

   — Command line tool to validate of ABNF grammars.

A list of all parts in the ISO/IEC 23009 series can be found on the ISO website.

Introduction

The conformance and reference software of the ISO/IEC 23009 series serves three main purposes:

— validation of the written specification of the Parts of the ISO/IEC 23009 series;

— clarification of the written specification of the Parts of the ISO/IEC 23009 series;

— conformance testing for checking interoperability for the various applications against the reference software which aims to be compliant with the ISO/IEC 23009 series.

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

Part 2: Conformance and reference software

1 Scope

This document specifies the conformance and reference software implementing the test vectors comprising media presentation descriptions, segments and combinations thereof in ISO/IEC 23009-1, and the corresponding software modules.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 19757-3, *Information technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation — Schematron*

ISO/IEC 23009-1:2019, *Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats*

ISO 8601-1, *Date and time — Representations for information interchange — Part 1: Basic rules*

ISO/IEC 23009-5:2017, *Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 5: Server and network assisted DASH (SAND)*

IETF RFC 7233:2014, *Hypertext Transfer Protocol (HTTP/1.1): Range Requests*

3 Terms, definitions, symbols and abbreviated terms

For the purpose of this document, the terms, definitions, symbols and abbreviated terms given in ISO/IEC 23009-1 apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

4 Media presentation conformance

4.1 Overview

A media presentation conforming to ISO/IEC 23009-1 obeys the rules for the media presentation description (MPD) and the segments referenced within the MPD. To verify the conformance of a media presentation, the following steps need to be completed:

— the conformance of the MPD according to Clause 5.

— the conformance of the segments, which includes the conformance of individual segments and representations, as well as the conformance of representations that are jointly provided in adaptation sets and periods. For details, refer to Clause 6.

The process of MPD and segment conformance checking is shown in Figure 1.

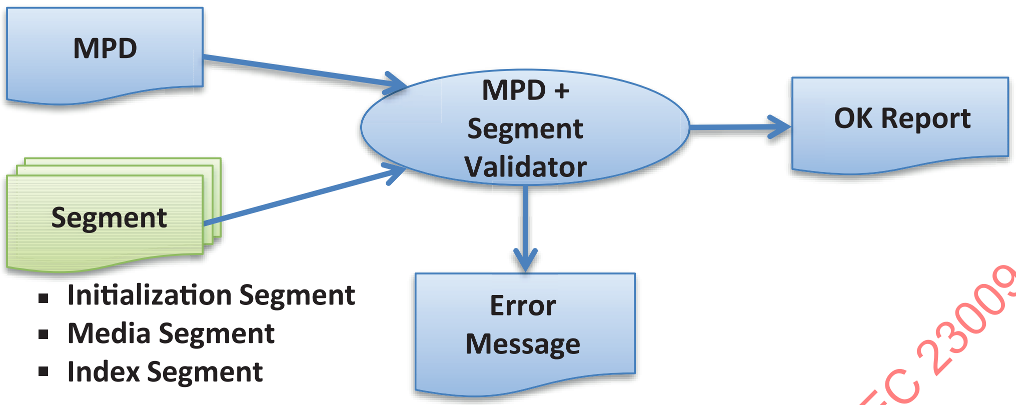

Figure 1 — MPD and segment validation

MPD + segment validator: gets as an input the MPD and segments referenced from within the MPD and performs the MPD and segment validation according to the rules defined in Annex A. On success, the output is an OK report; otherwise, an error message is provided.

4.2 Software tools

The following software tools are included:

— MPD conformance software;

— ISO BMFF segment validator;

— MPEG-2 TS segment validator;

— Conformance software for dynamic services;

— Dynamic media presentation emulator in Annex D.

All software tools are available at https://standards.iso.org/iso-iec/23009/-2/ed-3/en.

Additional supplemental software packages which may be used as sample DASH clients, sample segmentor and web-based conformance service are listed in Annex C.

Coverage of DASH features is discussed in Annex E.

5 MPD conformance

5.1 General

This clause specifies the MPD conformance checking and corresponding software modules which comprise the three steps depicted in Figure 2. Detailed means to perform MPD conformance checking are as specified in Annex A.

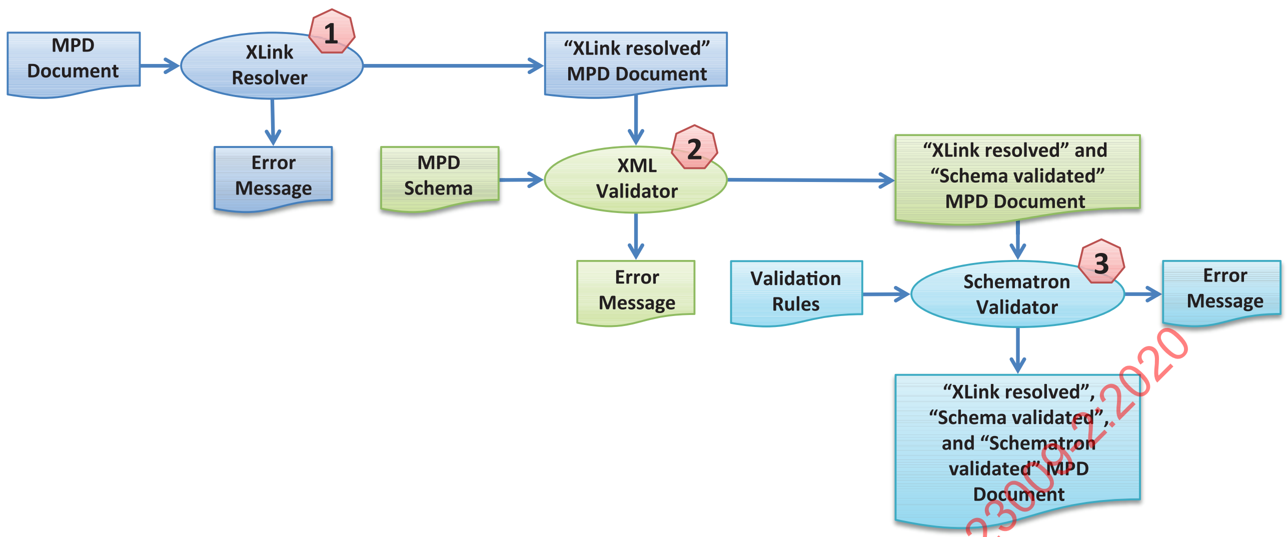
MPD Document
XLink Resolver
1
"XLink resolved" MPD Document
Error Message
MPD Schema
XML Validator
2
"XLink resolved" and "Schema validated" MPD Document
Error Message
Validation Rules
Schematron Validator
3
Error Message
"XLink resolved", "Schema validated", and "Schematron validated" MPD Document

Figure 2 — MPD conformance checking

Step 1 (XLink resolver): gets as an input an MPD document and resolves all W3C XLINK[10] attributes as defined in ISO/IEC 23009-1. If an error occurs, the corresponding error message shall be provided; otherwise, the XLink resolved MPD document is provided. The details for this step of the MPD conformance checking are defined in A.2.

Step 2 (XML validator): gets as an input an XLink-resolved MPD document and performs XML validation (i.e. well-formed and valid) against the MPD schema defined in W3C XML[11] and W3C XML SCHEMA[12]. If an error occurs, the corresponding error message shall be provided; otherwise, the XLink-resolved and schema-validated MPD document is provided. The details for this step of the MPD conformance checking are defined in A.3.

Step 3 (schematron validator): gets as an input an XLink-resolved and schema-validated MPD document and performs schematron validation as defined in ISO/IEC 19757-3 according to the rules defined in A.4.2. If an error occurs, the corresponding error message shall be provided; otherwise, the XLink-resolved, schema-validated, and Schematron-validated MPD document is provided. The details for this step of the MPD conformance checking are defined in A.4.

The following command may be used to validate an MPD document with the chain depicted in Figure 2. This requires Apache Ant$^{\text{TM}1)}$[3].

```
ant run -Dinput="filetovalidate.mpd"
```

The program outputs a message for each step. If an error occurs during one step, the following steps are not executed.

All conformance tools are available at https://standards.iso.org/iso-iec/23009/-2/ed-3/en.

5.2 Static MPD conformance

An MPD with `MPD@type="static"` shall comply with the rules in 5.1.

In addition, the availability of all resources in the MPD during the `MPD@availabilityStartTime` and the `MPD@availabilityEndTime` shall be checked. A function `remoteFileExists($url)` may be executed for each segment that is documented in the MPD. This function shall return true for all Segments in the MPD during the time interval defined by the `MPD@availabilityStartTime` and the `MPD@availabilityEndTime`. The following snippet shows an example for the `remoteFileExists` function written in PHP.

1) Apache Ant is the trade name of a product supplied by The Apache Software Foundation. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named.

```
    function remoteFileExists($url) {
        $curl = curl_init($url);

         //don't fetch the actual page, you only want to check the
        connection is ok curl_setopt($curl, CURLOPT_NOBODY, true);

         //do request
         $result = curl_exec($curl);

         $ret = false;

         //if request did not fail
         if ($result !== false) {
             //if request was ok, check response code
             $statusCode = curl_getinfo($curl, CURLINFO_HTTP_CODE);

             if ($statusCode == 200) {
             $ret = true;
             }
         }

             curl_close($curl);

             return $ret;
        }
```

5.3 Dynamic MPD conformance

5.3.1 General

To ensure that servers offering a dynamic DASH media presentation adhere to the timing requirements and that clients are able to properly consume the dynamic presentation without observing unavailable segments, the DASH conformance software is extended to include and address media presentations with **MPD**`@type='dynamic'`. In 5.3, the requirements which go beyond those for static DASH conformance are listed and a functional description of the conformance software implementation is provided.

The document is aligned with the guidelines for live services as documented in ISO/IEC TR 23009-3[1].

5.3.2 Background and requirements

5.3.2.1 MPD-specific checks

Clients make use of the information in an MPD. Based on an MPD that is fetched at a certain fetch time FT the client can determine the following information:

a) At any time, WT where WT ≥ FT, a client can determine:

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

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

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

   4) the URL of the latest segment that is available within this Period;

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

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

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

b) At any time, WT, all segments with availability start time at or before WT and availability end time at or before WT are accessible.

c) An updated MPD is made available on time, taking into account the minimum update period (MUP) of the MPD. Note that by not updating the MPD, the existing MPD is validated for another MUP time.

Detailed equations to derive segment availability times are provided in ISO/IEC 23009-1 and ISO/IEC TR 23009-3[1]. When providing an MPD, the content author ensures that the segment availability times can be derived.

5.3.2.2 MPD times

In order to use the same concepts with different timing and addressing schemes, the following values are introduced according to ISO/IEC 23009-1:

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

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

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

Assuming that the wall-clock time at the client is denoted as WT, the client can derive the following information.

5.3.2.3 General derivation

Using these times, the values from above can be derived as follows.

— The latest Period is the Period for which AST + PS + MD(1) ≤ WT where AST is the value of the `MPD@availabilityStartTime` and PS is the *PeriodStart* as defined in ISO/IEC 23009-1.

— The segment availability start time is obtained as

$$\text{SAST}(k) = \text{AST} + \text{PS} + \text{MST}(k) + \text{MD}(k)$$

— Within this Period, the latest segment available on the server is the segment at the position k^* which results in the greatest value for SAST(k^*) and at the same time is smaller than WT.

— The address of the latest segment can be derived using the position information k^*. The exact value depends on the addressing method.

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

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

It is further assumed that a segment at position k has an assigned earliest media presentation time EPT(k).

5.3.2.4 Requirements

When receiving an MPD, the DASH client is guaranteed that:

— Each segment at position k in this Period is available prior to the sum of its earliest presentation time and its duration; SAST(k) ≤ EPT(k) + MD(k).

— If each segment with segment number k is delivered starting at SAST(k) over a constant bitrate channel with bitrate equal to the value of the `@bandwidth` attribute, then each sample with presentation time *PT* shall be available at the client by the time given by PT + (AST + PS) + MBT + MD(k).

— Each segment in this Period shall be available at least until SAST(k) + TSB + MD(k).

— The MPD can be used to construct and request segments until media time FT + MUP. The greatest segment position k' that can be constructed by this MPD shall be the one that results in the greatest value for SAST(k') and at the same time is smaller than FT + MUP. Note that the latest segment may be significantly shorter in duration than MD(k).

In addition, updates of the MPD shall be consistent according to ISO/IEC 23009-1:2019, 5.4.

5.3.3 Dynamic conformance software design

5.3.3.1 Overview

The implementation of the conformance software for dynamic services requires the implementation of the requirements in 5.3.2.4. The process is split into two parts:

— the creation of a data set by recording/monitoring a dynamic service;

— the actual conformance checks implementing the checks of the requirements.

5.3.3.2 Data set creation

Two different cases are considered depending on the access bandwidth to the server.

In case 1, it is assumed that the conformance checker is operating with a very high (infinite) access bandwidth and minimal delay. Then the initial part is as follows.

a) Download and store the MPD and record the fetch time, FT, of the MPD.

b) Determine the following information.

 1) The smallest segment availability start time greater than FT of any Segment in any Representation referred to as $\mathrm{SAST}_{\mathrm{min}}$.

 2) The URLs of all segments available at $\mathrm{SAST}_{\mathrm{min}}$. If multiple segments are available at the same time, determine all of them.

c) For all segments with segment availability, end time greater than FT and smaller than $\mathrm{SAST}_{\mathrm{min}}$, generate the URL, issue an HTTP HEAD request, and record the HTTP header.

d) At time $\mathrm{SAST}_{\mathrm{min}}$ issue HTTP GET requests for all segments that become available at this time, record the HTTP headers and store the segments.

e) Go to a) and download the next MPD at FT = $\mathrm{SAST}_{\mathrm{min}}$.

f) Continue the process until manually stopped or the media presentation is ended.

In case 2, if the bandwidth connection is restricted, apply the following.

a) Download and store the MPD and record the fetch time, FT, of the MPD.

b) Determine the following information.

 1) The smallest segment availability start time greater than FT of any segment in any Representation referred to as $\mathrm{SAST}_{\mathrm{min}}$.

 2) The URLs of all segments available at $\mathrm{SAST}_{\mathrm{min}}$. If multiple segments are available at the same time, determine all of them.

c) For all segments with segment availability end time greater than FT and smaller than $\mathrm{SAST}_{\mathrm{min}}$, generate the URL, issue an HTTP HEAD request, and record the HTTP headers.

d) For selected representations, issue an HTTP GET for segments that are available at this time.

e) At time $SAST_{min}$ issue HTTP HEAD for all segments that become available at this time and record the HTTP headers.

f) Go to a) and download the next MPD at FT = $SAST_{min}$.

g) Continue the process until manually stopped or the media presentation is ended.

This initial operation results in a data set as follows:

a) a sequence of MPDs, each with a FT;

b) a set of Segment URLs with associated:

  1) SAST;

  2) fetch time for segment availability start time request (typically SAST);

  3) HTTP headers for segment availability start time request;

  4) segment availability end time (SAET);

  5) fetch time for segment availability end time request (typically just slightly before the SAET);

  6) HTTP headers for segment availability end time request;

c) at least for a selected set of representations, the corresponding segments.

From the sequence of the MPDs, a single new MPD can be generated with `MPD@type=static` that includes all segments.

5.3.3.3 Conformance checks

The conformance checks operate on the stored data set and include the following checks:

— the correctness of the sequence of MPDs as documented in ISO/IEC 23009-1:2019, 5.4.

— all MPD timing aspects and segment availability times as documented in ISO/IEC 23009-1:2019, 5.3.

— the static conformance checks using the single MPD according to 5.2.

5.4 Conformance checks for spatial relationship description

The conformance rules that relate to the spatial relationship description are provided in Table 1.

An MPD shall comply with the rules in Table 1 in accordance with ISO/IEC 23009-1.

Table 1 — Spatial relationship description conformance rules

Reference in ISO/IEC 23009-1: 2019, Annex H	Rule	Conformance check implementation
H.1	When every Adaptation Set in an MPD has an SRD descriptor, at least one such descriptor shall be a SupplementalProperty.	R19.1
H.1	An EssentialProperty or SupplementalProperty descriptor with @schemeIdUri equal to “urn:mpeg:dash:srd:2014” shall be the child element of an AdaptationSet or a SubRepresentation element.	R19.2
H.2	If an EssentialProperty or SupplementalProperty descriptor with @schemeIdUri equal to “urn:mpeg:dash:srd:2014” is present, then the @value attribute shall be present.	R19.3

Table 1 *(continued)*

Reference in ISO/IEC 23009-1: 2019, Annex H	Rule	Conformance check implementation
H.2	If an EssentialProperty or SupplementalProperty descriptor with @schemeIdUri equal to “urn:mpeg:dash:srd:2014” is present, then the @value attribute shallshall contain at least the mandatory comma separated parameters, i.e. `source_id`, `x`, `y`, `w`, `h`.	R19.4
H.2	If an EssentialProperty or SupplementalProperty descriptor with @schemeIdUri equal to “urn:mpeg:dash:srd:2014” is present, then each parameter value has to match the expected type format, i.e. non-negative integer in decimal representation.	R19.5
H.2	If an EssentialProperty or SupplementalProperty descriptor with @schemeIdUri equal to “urn:mpeg:dash:srd:2014” is present and the @value attribute contains the optional parameter `W` then the optional parameter `H` shall be present too.	R19.6
H.2	If an EssentialProperty or SupplementalProperty descriptor with @schemeIdUri equal to “urn:mpeg:dash:srd:2014” is present and the @value attribute contains the optional parameter `H` then the optional parameter `W` shall be present too.	R19.6
H.2	If an EssentialProperty or SupplementalProperty descriptor with @schemeIdUri equal to “urn:mpeg:dash:srd:2014” is present and the @value attribute contains the optional parameter `spatial_set_id` then the optional parameters `W` and `H` shall be present too.	R19.6
H.2	For a given `source_id` of the @value attribute, at least one of the EssentialProperty or SupplementalProperty in the containing Period shall specify the optional parameters `W` and `H`.	R19.7
H.2	For a given `source_id` of the @value attribute, if two SRD elements (indistinctively EssentialProperty or SupplementalProperty) explicitly specify a different pair of values for the optional parameters `(W,H)` then all the remaining SRD element shall explicitly specify a pair of values for `(W,H)` too.	R19.8
H.2	For a given `source_id` of the @value attribute, the values of x, w and `W` shall be such that, for each descriptor, the sum of `x` and `w` is smaller or equal to `W`.	R19.9a and R19.9b
H.2	For a given `source_id` of the @value attribute, the values of y, h and `H` shall be such that, for each descriptor, the sum of `y` and `h` is smaller or equal to `H`.	R19.10a and R19.10b

6 Segment conformance

6.1 Overview

Segment conformance requirements verify that the segments offered in the MPD conform to ISO/IEC 23009-1.

This includes the conformance requirements for:

— segments offered within one representation (for details, refer to 6.2);

— representations offered within one adaptation set (for details, refer to 6.3);

— segments offered in a dynamic media presentation (for details, refer to 6.4).

All conformance tools are available at https://standards.iso.org/iso-iec/23009/-2/ed-3/en.

6.2 Representation conformance

6.2.1 ISO base media file format

The representation conformance rules, as well as the implementation status of the conformance rules for ISO base media file format segments, are provided in Table 2.

Table 2 — Representation conformance rules for ISO base media file format

	Clause in ISO/IEC 23009-1: 2019	Rule	Conformance check implementation
1	6.1	Media Segment formats shall comply with the respective container formats (ISO BMFF and MPEG-2 TS).	implemented
2	6.2.1	The Initialization Segment shall not contain any media data with an assigned presentation time	implemented
3	6.2.1	A Media Segment shall contain a number of complete access units.	implemented
4	6.2.1	If it is the first Media Segment in the Representation, it shall contain only media streams that start with a SAP of type 1 or 2.	implemented
5	6.2.1	A Media Segment shall contain sufficient information to time-accurately present each contained media component in the Representation without accessing any previous Media Segment in this Representation provided that the Media Segment contains a SAP for each media stream.	implemented
6	6.2.3.2	A Media Segment shall specify all Media Presentation times relative to the start of the Period and compensated with the value of the @presentationTimeOffset. The presentation time in Media Segments shall be accurate to ensure accurate alignment of all Representations in one Period. a) earliest_presentation_time shall be equal to the sum of all temporally preceding subsegments in the representation. b) The duration of a subsegment indexed by an ‘sidx’ shall be equal to the sum of the durations of all the subsegments it indices.	implemented
7	6.3.2.1	A media data box containing data referenced by a movie fragment (‘moof’) box shall follow that movie fragment box and precede the next movie fragment box, if any, containing information about the same track.	implemented
8	6.3.2.1	For a Media Subsegment, the value of the reference_type field in the describing Segment Index (‘sidx’) box shall be set to 0.	implemented
9	6.3.2.3	If the Segment Index is provided, the Segment Index (‘sidx’) box in ISO/IEC 14496-12 shall be used.	implemented
10	6.3.2.4	If the Subsegment Index is provided, the Subsegment Index (‘ssix’) box in ISO/IEC 14496-12 shall be used.	implemented
11	6.3.3	The Initialization Segment shall contain an “ftyp” box, and a “moov” box.	implemented
12	6.3.3	It shall not contain any “moof” boxes.	implemented
13	6.3.3	The tracks in the “moov” box shall contain no samples (i.e. the entry_count in the “stts”, “stsc”, and “stco” boxes shall be set to 0).	implemented

Table 2 *(continued)*

	Clause in ISO/IEC 23009-1: 2019	Rule	Conformance check implementation
14	6.3.3	The “mvex” box shall be contained in the “moov” box. The “mvex” box also sets default values for the tracks and samples of the following movie fragments.	implemented
15	6.3.4.2	‘styp’ box, if present, shall carry ‘msdh’ as a compatible brand.	implemented
16	6.3.4.2	Each Media Segment shall contain one or more whole self-contained movie fragments. A whole, self-contained movie fragment is a movie fragment (‘moof’) box and a media data (‘mdat’) box that contains all the media samples that do not use external data references referenced by the track runs in the movie fragment box.	implemented
17	6.3.4.2	Each ‘moof’ box shall contain at least one track fragment.	implemented
18	6.3.4.2	The ‘moof’ boxes shall use movie-fragment relative addressing for media data that does not use external data references, the flag ‘default-base-is-moof’ shall be set, and data-offset shall be used, i.e. base-data-offset-present shall not be used.	implemented
19	6.3.4.2	Each ‘traf’ box shall contain a ‘tfdt’ box.	implemented
20	6.3.4.2	Each Media Segment may contain one or more ‘sidx’ boxes. If ‘sidx’ is present in a Media Segment, the first ‘sidx’ box shall be placed before any ‘moof’ box and the first Segment Index box shall document the entire Segment.	implemented
21	6.3.4.3	In each self-contained movie fragment, the movie fragment (‘moof’) box is immediately followed by its corresponding media data (‘mdat’).	implemented
22	6.3.4.3	Each Media Segment shall contain one or more ‘sidx’ boxes.	implemented
23	6.3.4.3	The first ‘sidx’ box shall be placed before any ‘moof’ box and shall document Subsegments that span the composition time of the entire Segment.	implemented
24	6.3.4.3	Each Media Segment shall carry ‘msix’ as a compatible brand.	implemented
25	6.3.4.4	The Subsegment Index box (‘ssix’) shall be present and shall follow immediately after the ‘sidx’ box that documents the same Subsegment. This immediately preceding ‘sidx’ shall only index Media Subsegments.	implemented
26	6.3.4.4	It shall carry ‘sims’ in the Segment Type box (‘styp’) as a compatible brand.	implemented
27	6.3.5.2	The Indexed Self-Initializing Media Segment shall carry ‘dash’ as a compatible brand.	implemented
28	5.3.5.2	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).	implemented

6.2.2 MPEG-2 transport stream

The representation conformance rules, as well as the implementation status of the conformance rules for MPEG-2 transport stream-based segments, are provided in Table 3.

Table 3 — Representation conformance rules for MPEG-2 transport stream

	Clause in ISO/IEC 23009-1: 2019	Rule	Conformance check implementation
1	6.1	Media Segment formats shall comply with the respective container formats (ISO BMFF and MPEG-2 TS).	implemented
2	6.2.2	The Initialization Segment shall not contain any media data with an assigned presentation time	implemented
3	6.2.3.1	A Media Segment shall contain a number of complete access units.	implemented
4	6.2.3.1	If it is the first Media Segment in the Representation, it shall contain only media streams that start with a SAP of type 1 or 2.	implemented for AVC streams (not verifiable on container level)
5	6.2.3.1	A Media Segment shall contain sufficient information to time-accurately present each contained media component in the Representation without accessing any previous Media Segment in this Representation provided that the Media Segment contains a SAP for each media stream (not verifiable on container level).	implemented
6	6.2.3.1	A Media Segment shall specify all Media Presentation times relative to the start of the Period and compensated with the value of the @presentationTimeOffset. The presentation time in Media Segments shall be accurate to ensure accurate alignment of all Representations in one Period. a) earliest_presentation_time shall be equal to the sum of all temporally preceding subsegments in the representation. b) The duration of a subsegment indexed by an 'sidx' shall be equal to the sum of the durations of all the subsegments it indices.	implemented
7	6.4.2.1	A subsegment shall contain complete access units for the indexed media stream (i.e. stream for which reference_ID equals PID)	implemented
8	6.4.2.2	PES packet starting at I_{SAU} shall contain only an integral number of access units and shall contain a PTS.	implemented
9	6.4.2.3	If the Segment Index is provided the Segment Index ('sidx') box in ISO/IEC 14496-12 shall be used for Segment Indexing.	implemented
10	6.4.2.3	reference_ID field of \`sidx\` box shall be the PID value of the indexed stream.	implemented
11	6.4.2.3	All media offsets within \`sidx\` boxes shall be to the first (sync) byte of a TS packet	implemented
12	6.4.2.4	If the Subsegment Index is provided, the Subsegment Index ('ssix') box in ISO/IEC 14496-12 shall be used for indexing byte ranges within a subsegment	implemented
13	6.4.2.4	All media offsets within 'ssix' boxes shall be to the first (sync) byte of a TS packet	implemented
14	6.4.3.2	An Initialization Segment shall be a valid MPEG-2 TS, conforming to ISO/IEC 13818-1.	implemented
15	6.4.3.2	The concatenation of an Initialization Segment with any Media Segment shall have the same presentation duration as the original Media Segment.	implemented

Table 3 *(continued)*

	Clause in ISO/IEC 23009-1: 2019	Rule	Conformance check implementation
16	6.4.3.2	The Initialization Segment shall contain mandatory untimed initialization information as defined in ISO/IEC 23009-1:2019, 6.4.3.1 in this order: a) PAT; b) PMT; c) If MPEG-2 conditional access is used, ECM.	implemented (w/o ECM)
17	6.4.4.2	Media Segments shall contain complete MPEG-2 TS packets	implemented
18	6.4.4.2	Media Segments shall contain exactly one program	implemented
19	6.4.4.2	All time-varying initialization information shall be present between I_{SAP} and I_{SAU} and/or in the Index Segment, if present.	implemented
20	6.4.4.2	No Media Segment shall depend on initialization information appearing in any preceding Media Segment.	implemented
21	6.4.4.3	All information necessary for decrypting, or locating information required to decrypt, the encrypted TS packets in a (Sub)Segment shall be present before the encrypted packet(s) to which they apply, either in the same (Sub)Segment, and/or in the Initialization Segment (if used).	implemented (as much as possible)
22	6.4.4.4	A Self-initializing Media Segment shall contain, at the least, all mandatory untimed and timed initialization information as defined in ISO/IEC 23009-1:2019, 6.4.3.1.	implemented
23	6.4.5	A Bitstream Switching Segment shall be a valid MPEG-2 TS, conforming to ISO/IEC 13818-1.	implemented
24	6.4.5	A Bitstream Switching Segment, when concatenated with any Media Segment, shall not alter the Media Presentation timeline for the corresponding Media Segment.	implemented
25	6.4.5	If initialization information is carried within a Bitstream Switching Segment, it shall be identical to the one in the Initialization Segment, if present, of the Representation.	implemented
26	6.4.6.2	A Single Index Segment indexes exactly one Media Segment and is defined as follows: a) Each Single Index Segment shall begin with an ‘styp’ box, and the brand ‘sisx’ shall be present in the ‘styp’ box. b) Each Single Index Segment shall contain one or more Segment Index boxes which index one Media Segment.	implemented
27	6.4.6.2	If present, the ‘ssix’ shall follow the ‘sidx’ box that documents the same Subsegment without any other ‘sidx’ preceding the ‘ssix'.	implemented
28	6.4.6.2	If present, ‘pcrb’ shall follow the ‘sidx’ box that documents the same Subsegments.	implemented
29	6.4.6.3	Each Representation Index Segment shall begin with an ‘styp’ box, and the brand ‘risx’ shall be present in the ‘styp’ box.	implemented

Table 3 *(continued)*

	Clause in ISO/IEC 23009-1:2019	Rule	Conformance check implementation
30	6.4.6.3	The Segment Index for each Media Segments is concatenated in order, preceded by a single Segment Index box that indexes the Index Segment. This initial Segment Index box shall have one entry in its loop for each Media Segment, and each entry refers to the Segment Index information for a single Media Segment.	implemented
31	6.4.6.4	It shall be either a Single Index Segment or a Represent tion Index Segment.	implemented
32	6.4.6.4	The Subsegment Index box ('ssix') shall be present and shall follow immediately after the 'sidx' box that documents the same Subsegment.	implemented
33	6.4.6.4	The value of the reference_type field shall be equal to 0 for this Subsegment in this immediately preceding Segment Index ('sidx') box.	implemented
34	6.4.6.4	If the 'pcrb' box is present, it shall follow 'ssix'.	implemented
35	6.4.6.4	It shall carry 'ssss' in the Segment Type box ('styp') as a compatible brand.	implemented
36	7.1	The Media Presentation shall be provided such that no mismatch between the media stream and the MPD occurs.	implemented

6.3 Adaptation set conformance

6.3.1 ISO base media file format

The adaptation set conformance rules, as well as the implementation status of the conformance rules for ISO base media file format segments, are provided in Table 4.

Table 4 — Adaptation set conformance rules for ISO base media file format

	Clause in ISO/IEC 23009-1:2019	Rule	Conformance check implementation
1	7.2.2	If a Segment Index is present in a Media Segment of one Representation within an Adaptation Set, then the following shall hold. a) The order of Segment Index boxes for multiple media streams induces an ordering on the media content components equal to the order in which a Segment Index box for a media stream for each component first appears. This ordering shall be the same for all Segments of all Representations of an Adaptation Set. As a consequence, if there is a Segment Index for a media content component in one Segment, there shall be a Segment Index for that media component in all Segments in this Adaptation Set. b) Non-indexed media streams in all Representations of an Adaptation Set shall have the same access unit duration.	implemented

Table 4 *(continued)*

	Clause in ISO/IEC 23009-1: 2019	Rule	Conformance check implementation
2	7.3.3.2	As a consequence of @bitstreamSwitching being set to ‘true’, the following conditions are satisfied. a) The track IDs for the same media content component are identical for each Representation in each Adaptation Set. b) The conditions required for setting the @segmentAlignment attribute to a value other than 'false' for the Adaptation Set are fulfilled. c) The conditions required for setting the @startWithSAP attribute to 2 for the Adaptation Set, or the conditions required for all Representations within the Adaptation Set to share the same value of @mediaStreamStructureId and setting the @startWithSAP attribute to 3 for the Adaptation Set, are fulfilled.	implemented
3	7.3.4	If a SubRepresentation element is present in a Representation in the MPD and the attribute SubRepresentation@level is present, then the Media Segments in this Representation shall conform to a Sub-Indexed Media Segment as defined in ISO/IEC 23009-1:2019, 6.3.4.4. The Initialization Segment shall contain the Level Assignment (‘leva’) box.	implemented

6.3.2 MPEG-2 transport stream

The adaptation set conformance rules, as well as the implementation status of the conformance rules for ISO base media file format segments, are provided in Table 5.

Table 5 — Adaptation set conformance rules for MPEG-2 transport stream

	Clause in ISO/IEC 23009-1: 2019	Rule	Conformance check implementation
1	7.1	The Media Presentation shall be provided such that no mismatch between the media stream and the MPD values occurs. If it does, the value in the media stream itself takes precedence over values expressed in the MPD, especially when used in the media decoding process.	implemented

Table 5 *(continued)*

	Clause in ISO/IEC 23009-1: 2019	Rule	Conformance check implementation
2	7.2.2	If a Segment Index is present in a Media Segment of one Representation within an Adaptation Set, then the following shall hold. a) The order of Segment Index boxes for multiple media streams induces an ordering on the media content components equal to the order in which a Segment Index box for a media stream for each component first appears. This ordering shall be the same for all Segments of all Representations of an Adaptation Set. As a consequence, if there is a Segment Index for a media content component in one Segment, there shall be a Segment Index for that media component in all Segments in this Adaptation Set. b) Non-indexed media streams in all Representations of an Adaptation Set shall have the same access unit duration.	implemented
3	7.4.3.2	If the @segmentAlignment attribute is not set to ‘false’: a) the Media Segment shall contain only complete PES packets; b) the first PES packet shall contain a PTS timestamp.	implemented
4	7.4.3.3	If the @subsegmentAlignment flag is not set to 'false', a Subsegment shall contain only complete PES packets for each PID.	implemented
5	7.4.3.3	If the @subsegmentAlignment flag is not set to 'false', the first PES packet from each elementary stream shall contain a PTS.	implemented
6	7.4.3.4	If the @bitstreamSwitching is set to true, then for any two Representations, X and Y, within the same Adaptation Set, concatenation of Media Segment i of X, Concatenation Segment of Representation Y, and Media Segment i+1 of Representation Y shall be a MPEG-2 TS conforming to ISO/IEC 13818-1.	implemented
7	7.4.3.4	If the @bitstreamSwitching is set to true, then the conditions required for setting the @startWithSAP attribute to 2 for the Adaptatation Set or required for all Representations within the Adaptation Set share the same value of @mediaStreamStructureId and setting the @startWithSAP attribute of the Adapaton Set 3, are fulfilled.	implemented
	7.4.3.4	If the @bitstreamSwitching is set to true, then the conditions required for setting the @segmentAlignment attribute not set to 'false' for the Adaptation Set are fulfilled.	implemented
	7.4.3.4	If the @bitstreamSwitching is set to true, then PCR shall be present in the Segment prior to the first byte of a TS packet payload containing media data, and not inferred from the ‘pcrb’ box.	implemented
	7.4.4	The Subsegment Index box shall contain at least one entry for the value of SubRepresentation@level and for each value provided in the SubRepresentation@dependencyLevel.	implemented

6.4 Dynamic media presentation conformance

With the dynamic MPD conformance checks defined in 5.3, no specific segment conformance for dynamic services are necessary. The static segment conformance is sufficient to also address services with `MPD@type=dynamic`.

7 Profile specific conformance

NOTE Profile-specific conformance for MPEG-2 TS main profile, ISO base Media File Format Extended Live profile, ISO Base Media File Format Extended On Demand profile, and ISO Base Media File Format Common profile are missing.

7.1 ISO base media file format on demand profile

The profile-specific conformance rules, as well as the implementation status of the conformance rules for the ISO base media file format on-demand profile, are provided in Table 6.

Table 6 — ISO base media file format on-demand profile rules

	Clause in ISO/IEC 23009-1:2019	Rule	Conformance check implementation
1	8.3.3	All Segment Index ('sidx') and Subsegment Index ('ssix') boxes shall be placed before any Movie Fragment ('moof') boxes.	implemented
2	8.4.3	Media Segments containing multiple Media Components shall comply with the formats defined in ISO/IEC 23009-1:2019, 6.3.4.3, i.e. the brand 'msix'.	implemented
3	8.4.3	In Media Segments, all Segment Index ('sidx') and Subsegment Index ('ssix') boxes shall be placed before any Movie Fragment ('moof') boxes.	implemented

7.2 ISO base media file format live profile

The profile-specific conformance rules, as well as the implementation status of the conformance rules for the ISO base media file format live profile, are provided in Table 7.

Table 7 — ISO base media file format live profile rules

	Clause in ISO/IEC 23009-1:2019	Rule	Conformance check implementation
1	8.4.3	Media Segments containing multiple Media Components shall comply with the formats defined in ISO/IEC 23009-1:2019, 6.3.4.3, i.e. the brand 'msix'.	implemented
2	8.4.3	In Media Segments, all Segment Index ('sidx') and Sub-segment Index ('ssix') boxes shall be placed before any Movie Fragment ('moof') boxes.	implemented

7.3 ISO base media file format main profile

The profile-specific conformance rules, as well as the implementation status of the conformance rules for the ISO base media file format main profile, are provided in Table 8.

Table 8 — ISO base media file format main profile rules

	Clause in ISO/IEC 23009-1:2019	Rule	Conformance check implementation
1	8.5.3	At least one SAP of type 1 to 3, inclusive, shall be present for each track in each subsegment	implemented
2	8.5.3	In Media Segments, all Segment Index ('sidx') and Subsegment Index ('ssix') boxes shall be placed before any Movie Fragment ('moof') boxes.	implemented
3	8.5.3	Each Media Segment of the Representations not having @startWithSAP present or having @startWithSAP value 0 or greater than 3 shall comply with the formats defined in ISO/IEC 23009-1:2019, 6.3.4.3, i.e. the brand 'msix'.	implemented

7.4 MPEG-2 transport stream simple profile

The profile-specific conformance rule, as well as the implementation status of the conformance rules for the MPEG-2 transport stream simple profile, are provided in Table 9.

Table 9 — MPEG-2 transport stream simple profile rules

	Clause in ISO/IEC 23009-1:2019	Rule	Conformance check implementation
1	8.7.3	PSI information, including versions, shall be identical within all Representations contained in an AdaptationSet	implemented
2	8.7.3	If MPEG-2 conditional access framework is used, same ECM shall be valid for the whole Subsegment, or for the whole Segment, if Index Segment is not present.	Specific to a conditional access system, cannot be tested generically.
3	8.7.3	For an Index Segment, any single Segment Index ('sidx') box may either reference media, or other 'sidx', but the same 'sidx' box may not reference both.	implemented

8 Conforming test vectors

Details on conforming test vectors are specified in Annex B.

9 DASH access engine reference software

9.1 General

This clause defines the DASH access engine reference software according to ISO/IEC 23009-1. In this clause, this reference software is referred to as libdash[8].

NOTE Reference software implements requirements of a specification and typically serves the following purposes, (1) validation of the written specification, (2) clarification of the written specification, and (3) conformance testing for checking interoperability for the various applications against the reference software.

9.2 libdash overview

ISO/IEC 23009-1 defines representation formats for the MPD and segment formats for MPEG-2 transport stream (M2TS) and ISO base media file format (ISOBMFF). The MPD is based on XML and reflects the DASH data model: providing HTTP URLs to downloadable segments. The adaptation logic (the component of a client which determines which segment to download based on the clients' context) is deliberately excluded from the specification and left open for industry competition.

libdash focuses on the MPD and segments identified using HTTP URLs, and provides two features:

— an object-oriented interface into the MPD;

— the download of segments via HTTP (on request).

In particular, libdash provides means to access the information within the MPD and enables scheduling the download of segments described by the MPD.

libdash implements the full profile of ISO/IEC 23009-1.

libdash does not include adaptation logic and does not provide any segment- or codec-specific features.

9.3 libdash-enabled example system

Figure 3 shows a libdash-enabled example system comprising a server (hosting the MPD and segments of various representations described within the MPD) and client (requesting segments according to the MPD). The delivery of the MPD and client components such as the DASH streaming control (including adaptation logic) and the media player are specified within ISO/IEC 23009-1.

libdash provides interfaces for the DASH streaming control (including adaptation logic) and the media player to access the MPD and downloadable media segments. The download order of such media segments is not handled by libdash as this is left open to the other components.

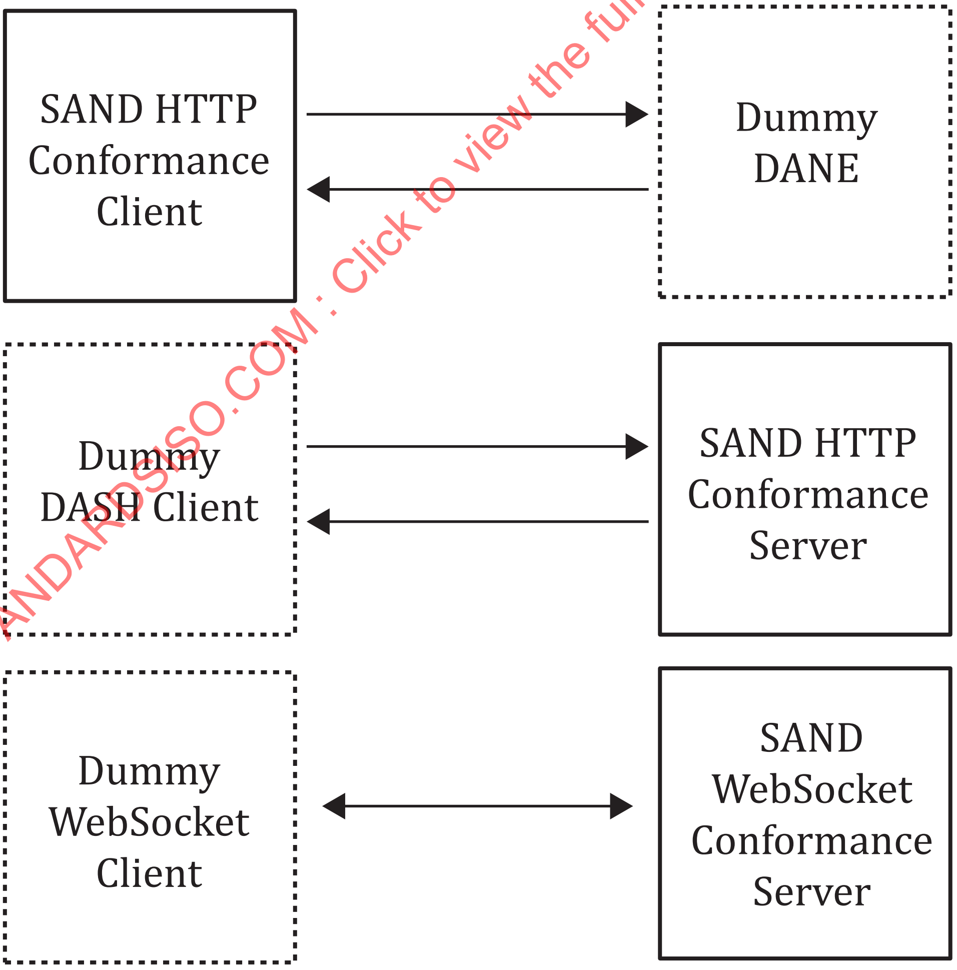
SAND HTTP Conformance Client
Dummy DANE
Dummy DASH Client
SAND HTTP Conformance Server
Dummy WebSocket Client
SAND WebSocket Conformance Server

Figure 3 — libdash-enabled example system

In an example deployment, a server may provide segments in several bitrates and resolutions. The client initially receives an MPD through libdash, which provides a convenient object orient interface to that MPD. Based on that information, the client can download individual media segments through libdash at any point in time. For example, varying bandwidth conditions can be handled by switching to the corresponding quality level at segment boundaries in order to provide a smooth streaming experience. This adaptation is not part of libdash and is left open to the application using libdash.

9.4 libdash availability

libdash is available at https://standards.iso.org/iso-iec/23009/-2/ed-3/en.

10 Conformance software for ISO/IEC 23009-4

10.1 General

In each of these cases, output media segments are saved to the local disk, and the MPD is transformed appropriately and also saved to the local disk. However, the input MPD and media segments may come from either the local disk or from URLs.

In order to compile the utility, open Eclipse and import "mpdCrypto" as an “Existing project”. Eclipse JDT will compile the needed .class files automatically.

10.2 Design limitations and assumptions

— A single MPD is processed.

— Only the first `Representation` in the first `AdaptationSet` of the first `Period` is processed.

— Segments are assumed to be listed in a single `SegmentList`.

— A single instance of `ContentProtection` descriptor of supported type is assumed in encrypted MPDs.

— Support for validating MPDs via Schematron is coming at a later date.

— All of the operations are performed serially and synchronously.

10.3 Usage

mpdEncryptionUtil [-a <authentication-scheme>] -d | -e <encryption-scheme> -i <inputfile.mpd> [--no-validate-schema] -o <output-dir>

-a,--authenticate <authentication-scheme>

Generate authenticity tags for cleartext segments, in addition to encrypting them, via one of supported authentication schemes, "sha256" and "hmac-sha1". Default value is "sha256".

-d,--decrypt

Decrypt encrypted segments, automatically validating them, if authentication information is present.

-e,--encrypt <encryption-scheme>

Encrypt cleartext segments via one of supported encryption schemes, "aes128-cbc", "aes128-gcm". Default value is "aes128-cbc".

-i,--input-file <inputfile.mpd>

MPD file path or URL.

--no-validate-schema

Whether to skip validation of MPD via DASH schema.

-o,--output-dir <output-dir>

Directory to which encrypted/decrypted segments will be written.

11 Conformance for ISO/IEC 23009-5 server and network-assisted DASH (SAND)

11.1 Conformance rules

The conformance rules related to ISO/IEC 23009-5 as well as the implementation status of the conformance rules for SAND are provided in Table 10.

An MPD shall comply with the rules 5.G.1 to 5.H.3 in Table 10 in accordance with ISO/IEC 23009-5.

SAND messages shall comply with the rules 5.A.1 to 5.B.6 in Table 10 in accordance with ISO/IEC 23009-5.

Table 10 — Conformance rules for SAND

#	Object	Reference in ISO/IEC 23009-5:2017	Normative statements from ISO/IEC 23009-5:2017	Conformance rule	Status
5.A.1	messageType	Table 9	When using XML representation format, this field shall (messageType) be represented by the tag name of the message element.	SAND schema	covered
5.A.2	SAND message representation format	7	While SAND network elements may implement additional data representation formats, they shall at least support SAND messages in XML format.	SAND Schema	covered
5.A.4	SAND message representation format	7	While SAND network elements may implement additional data representation formats, they shall at least support SAND messages in XML format.	SAND Schema	covered
5.A.5	DCOR1 SAND schema fix			SAND Schema	covered
5.A.6	AbsoluteDeadline	6.4.4.1	AbsoluteDeadline message shall not be sent using the HTTP POST method.	SAND Schema	covered
5.A.7	MPDValidityEndTime	6.5.4.3	mpdUrl and mpd parameters are mutually exclusive and one of the two shall be present in any MPDValidityEndTime message.	SAND Schema	covered
5.A.7	ResourceStatus	Table 11	Either Representation ID or Base URL shall be used.	SAND Schema	covered
5.B.1	SharedResource Assignment	6.5.3.3	The validityTime attribute in the SAND message envelope shall be present for the SharedResourceAssignment message as it allows the DASH client to discover for how long the resource assignment is available.	SAND schematron	covered
5.B.3	QoSInformation	6.5.7.3	At least one of the above parameters (@gbr, @mbr, @delay, @pl) shall exist in the message.	SAND schematron	covered
5.B.4	bytes	Table 14	The string shall conform to the byte-range-set according to IETF RFC 7233:2014, subclause 2.1.	SAND Schematron	covered
5.B.5	AvailabilityTimeOffset	Table 21	Either Representation ID or Base URL shall be used.	SAND Schematron	covered
5.B.6	Throughput	Table 19	Either Representation ID or Base URL shall be used.	SAND Schematron	covered
5.C.1	messageType	Table 2	When using the HTTP header transport of 8.2.3, this field (messageType) shall be represented as the header name.	HTTP header syntax checker	covered
5.C.2	bytes	Table 14	The string shall conform to the byte-range-set according to IETF RFC 7233:2014, subclause 2.1.	HTTP header syntax checker	covered

Table 10 *(continued)*

#	Object	Reference in ISO/IEC 23009-5:2017	Normative statements from ISO/IEC 23009-5:2017	Conformance rule	Status

5.C.3	Attaching a message to requests for media	8.2.3	The header value provides the message data. The header value shall conform to the following ABNF:
			`sand-message-value = sand-object`
			`sand-object = sand-attr-or-list *( "," sand-attr-or-list )`
			`sand-attr-or-list = sand-attribute / sand-list`
			`sand-list = "[" sand-object *( ";" sand-object ) "]"`
			`sand-attribute = sand-attribute-name "=" sand-value`
				HTTP header syntax checker	covered

5.C.4	Attaching a message to requests for media	8.2.3	DATETIME shall follow the ISO 8601-1 format, with a dot and not a comma for fractions of second to avoid confusion with comma separating key=value pairs.	HTTP header syntax checker	covered
5.C.5	Attaching a message to requests for media	8.2.3	If attributes defined for the envelope or attributes defined as common to all messages are included, they shall appear first in the message, and shall be provided as key=value pairs.	HTTP header syntax checker	covered
5.D.2	Transport Protocol to carry SAND Messages	8.1	This clause defines HTTP as the minimum transport protocol that shall be at least supported by SAND enabled elements.	SAND server	covered
5.D.3	Transport Protocol to carry SAND Messages	8.1	The following table summarizes which HTTP usages shall be supported (in bold in the table) by a SAND element or may be optional depending on the nature of the SAND message.	SAND server	covered
5.D.4	Sending a message directly to a DANE	8.2.2	The `Content-Type` shall be `application/sand+xml`, as defined in Annex C The message document itself is included as the body part of the HTTP request.	SAND server	covered
5.E.1	Transport Protocol to carry SAND Messages	8.1	This clause defines HTTP as the minimum transport protocol that shall be at least supported by SAND enabled elements.	SAND client	covered
5.E.2	Transport Protocol to carry SAND Messages	8.1	The following table summarizes which HTTP usages shall be supported (in bold in the table) by a SAND element or may be optional depending on the nature of the SAND message.	SAND client	covered

5.E.3	Assistance	8.3.2	Upon receiving an HTTP entity that contains the SAND header field in its entity head, the DASH client issues a GET request to the indicated element to receive the SAND message. The following ABNF syntax for the header field shall be used:
			`SAND-header-field = "MPEG-DASH-SAND" ":" element-address`
			`element-address = absolute-URI`
				SAND client	covered

5.F.1	WebSocket messages	10.2.3	Each WebSocket message shall contain a valid SAND message compliant with the SAND message XML schema.	WS SAND server and client	covered
5.G.1	`urn:mpeg:dash:sand:2015:channel:http:2016`	Table 27	In this case, the `@endpoint` of the `sand:Channel` shall be a valid HTTP-URL.	MPD schema	covered
5.H.1	`urn:mpeg:dash:sand:channel:header:2016`	Table 27	In this case, the `@endpoint` of the `sand:Channel` shall not be present.	MPD schematron	covered
5.H.2	`urn:mpeg:dash:sand:channel:websocket:2016`	10.1	In this case, the `@endpoint` of the `sand:Channel` shall be a valid WebSocket URI as specified in RFC 6455:2011, Clause 3.	MPD schematron	covered

Table 10 *(continued)*

#	Object	Reference in ISO/IEC 23009-5:2017	Normative statements from ISO/IEC 23009-5:2017	Conformance rule	Status
5.H.3	`urn:mpeg:dash:sand:channel:2016`	Table 29	In this case, the `Reporting@value` attribute shall contain the value of a `sand:Channel@id` attribute present in the MPD.	MPD Schematron	covered
	messageID	Table 2	Among messages with same sender and same messageType, the message with highest messageID value shall take precedence over the others.	No rule	N/A
	messageID	Table 2	The maximum value for messageId is decided by senders and shall be high enough for receivers to easily identify which message shall take precedence even when messageID values have looped back to 0.	No rule	N/A
	bandwidth	Table 5	If playback rate is not 1, this bandwidth value shall be modified accordingly.	No rule	N/A
	AcceptedAlternatives	6.4.3.1	As such message is sent at the same time as a DASH segment request, transport protocol defined in subclause 8.2.3 (use of header extensions) shall be used.	No rule	N/A
	MaxRTT	6.4.5.1	If MaxRTT message is sent with HTTP POST method, it shall be intended for every segment request during validity time.	No rule	N/A
	clientID	Table 16	This field shall use the same value as the senderId of the SharedResourceAllocation message sent by the client.	No rule	N/A
	SAND message representation format	7	This clause defines XML as the format for SAND messages data representation format. While SAND network elements may implement additional data representation formats (such as JSON), they shall at least support SAND messages in XML format.	No rule	N/A
	Attaching a message to requests for media	8.2.3	For SAND messages defined in another namespace than the MPEG namespace (urn:mpeg:dash:schema:sandmessage:2016), the namespace shall be present in the SAND message name with colon characters replaced with hyphens. For instance: SAND-urn-my-example-com-MyMessage.	No rule	N/A
	Signalling of SAND communication channel	9.1	The `sand:Channel@schemeIdUri` specifies which protocol the DASH client shall use with this SAND channel. The table below lists the mandatory protocols.	No rule	N/A
	`urn:mpeg:dash:sand:2015:channel:http:2016`	Table 28	The identifier indicates that the DASH client shall use the protocol defined in ISO/IEC 23009-5:2017, subclause 8.1.1, that is, the SAND messages are transmitted via HTTP POST request inside the body of the request.	No rule	N/A
	`urn:mpeg:dash:sand:channel:header:2016`	Table 28	The identifier indicates that the DASH client shall use the protocol defined in ISO/IEC 23009-5:2017, subclause 8.1.2, that is, the SAND messages are transmitted via HTTP header extension of HTTP requests for media segments and MPD.	No rule	N/A
	Signalling of SAND communication channel	9.1	DASH clients which implement SAND shall support signalling of SAND channel via HTTP header and MPD.	No rule	N/A
	`urn:mpeg:dash:sand:channel:websocket:2016`	10.1	The identifier indicates that the DASH client shall use the WebSocket Protocol as specified in ISO/IEC 23009-5:2017, 10.2 WebSocket Protocol.	No rule	N/A
	Signalling via the MPD	10.2.2	The `@schemaIdUri` attribute of the `sand:Channnel` element shall be `urn:mpeg:dash:sand:channel:websocket:2016` while the `@endpoint` attribute shall contain a valid WebSoscket URI as defined in RFC 6455:2011, Clause 3.	No rule	N/A
	`urn:mpeg:dash:sand:channel:2016`	Table 30	The identifier indicates that the DASH client shall use a SAND channel to report the metrics defined in ISO/IEC 23009-1.	No rule	N/A

Table 10 *(continued)*

#	Object	Reference in ISO/IEC 23009-5:2017	Normative statements from ISO/IEC 23009-5:2017	Conformance rule	Status
	SharedResource Allocation Allocation Strategies	C.1	Implementation of these strategies shall follow the normative algorithms described in this annex. When DASH clients do not agree on the same strategy, it is up to the DANE to decide which strategy to apply.	No rule	N/A

11.2 Software

11.2.1 Design and architecture

The conformance of ISO/IEC 23009-5 comprises two modules, namely the SAND Conformance Client and the SAND Conformance Server. The SAND Client validates the SAND protocol and the SAND messages sent by the HTTP server simulating a DANE (called Dummy DANE in Figure 4). For instance, a DANE can insert a URL pointing to a SAND message in the HTTP header of a segment response. In this scenario, the SAND Client verifies that the SAND HTTP header is present. If present, it sends a HTTP request to that URL. Upon reception, it checks the received SAND messages against the SAND XML schema and schematron. Conversely, the SAND Server validates the SAND protocol and the SAND messages sent by an HTTP client simulating a DASH Client.

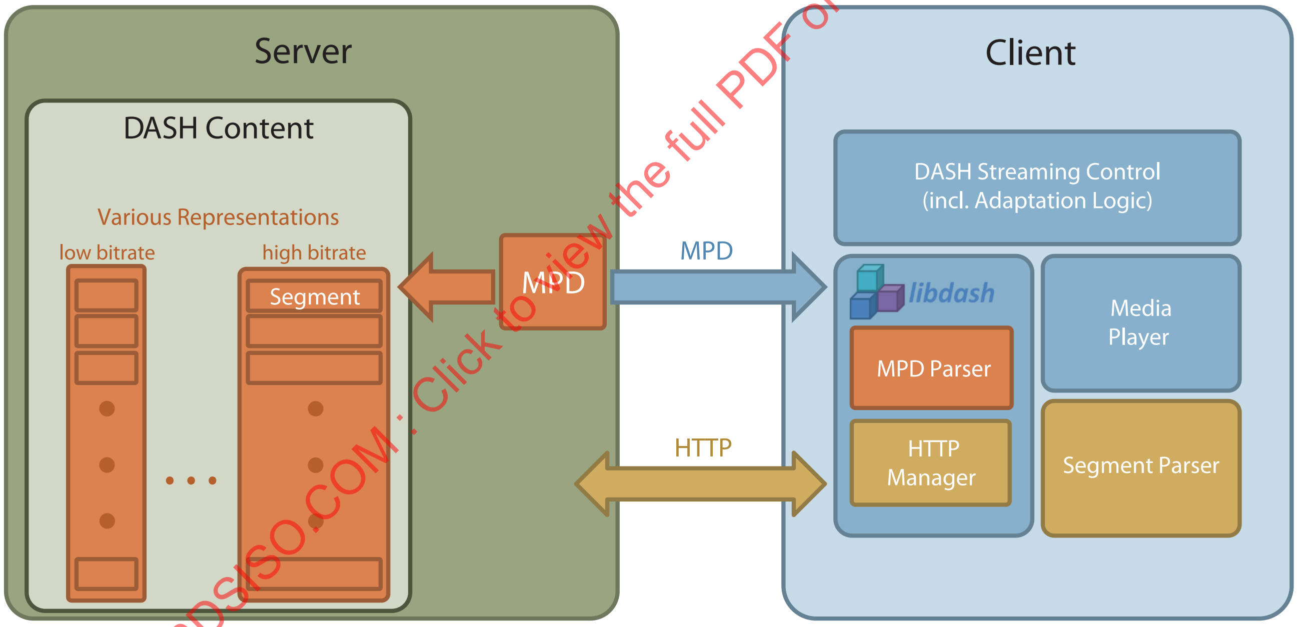

Figure 4 — Scope of conformance software for ISO/IEC 23009-5

11.2.2 Usage

11.2.2.1 SAND HTTP conformance client

A complete documentation is provided in the repository with dependencies and installation procedure. Basic usage information is provided here.

```
Usage: sand_client.py [OPTIONS]


  Execute the request and the validation of the received HTTP response.


Options:
  -u, --url_to_request TEXT       DANE URL to request, e.g. http://mydane.com
  -p, --protocol [assistance|enforcement|error]
                                  Protocols to carry PER messages specified in
                                  ISO/IEC 23009-5
  --help                          Show this message and exit.
```

Example

```
python sand_client.py -u http://my-dane.com/trigger-assistance -p assistance
```

Example of valid assistance PER transport

```
INFO:root:[TEST] header OK|mpeg-dash-sand header found in the response
INFO:root:[TEST] URL OK|The provided URL is valid|http://my-dane.example.com/messages/
MpdValidityEndTime1.xml
DEBUG:requests.packages.urllib3.connectionpool:Starting new HTTP connection (1): my-dane.
example.com
DEBUG:requests.packages.urllib3.connectionpool:http://my-dane.example.com:80 "GET /
messages/MpdValidityEndTime1.xml HTTP/1.1" 200 307
INFO:root:[TEST] content-type header|OK|Valid content type|application/sand+xml
INFO:root:[TEST] HTTP response code|OK|value=200
INFO:root:[TEST] message|OK|XML message format valid
INFO:root:[RESULT] Success
```

Example of invalid error PER transport

```
INFO:root:[TEST] content_type header|KO|Wrong content type|expected=application/
sand+xml|used=text/html; charset=utf-8
INFO:root:[TEST] HTTP response code|KO|value=200|regex=4[0-9]{2}
Start tag expected, '<' not found, line 1, column 1
INFO:root:[TEST] message|KO|XML message format invalid
INFO:root:[RESULT] Failure
```

11.2.2.2 SAND HTTP conformance server

A complete documentation is provided in the repository with dependencies and installation procedure. Basic usage information is provided here.

```
python sand_server.py run --port tcp_port
```

By default, the server listens on port 5000.

Endpoints

`/header`

This endpoint validates that a DASH client sends a SAND message via the HTTP header as specified by ISO/IEC 23009-5 SAND.

HTTP response — Failed validation

```
Report for SAND headers conformance:
Sand-Maxrtt: PASSED
```

HTTP response — Failed validation

```
Report for SAND headers conformance:
Sand-Maxrtt: FAILED
   Unexpected sand-attribute name 'maxRT'. Stopping parsing.
```

`/metrics`

This endpoint validates that a DASH client sends a metric message via the HTTP POST method as specified by ISO/IEC 23009-5 SAND.

Example of valid request

```
POST /metrics HTTP/1.1

Accept: application/json, */*

Accept-Encoding: gzip, deflate

Connection: keep-alive

Content-Length: 296

Content-type: application/sand+xml

Host: sand-conf-server.my-example-server.com

User-Agent: HTTPie/0.9.9


<?xml version="1.0" encoding="UTF-8"?>
<SANDMessage xmlns="urn:mpeg:dash:schema:sandmessage:2016" senderId="abc1234"
generationTime="2016-02-21T11:20:52-08:00">

  <TcpList messageId="1234">

    <TcpConnection tcpid="143" dest="542" topen="2016-04-21T14:54:52-08:00"/>

  </TcpList>

</SANDMessage>


HTTP/1.1 200 OK

Connection: keep-alive

Content-Length: 12

Content-Type: text/html; charset=utf-8

Date: Tue, 18 Jul 2017 12:35:27 GMT

Server: gunicorn/19.6.0

Via: 1.1 vegur


[RESULT][OK]
```

Server log

```
INFO:root:[TEST][OK] HTTP method (POST)

INFO:root:[TEST][OK] Content-Type (application/sand+xml)

INFO:root:[TEST][OK] SAND message validation
INFO:root:[RESULT][OK]
```

Example of invalid request (HTTP method, content-type header, message format failed)

```
PUT /metrics HTTP/1.1

Accept: application/json, */*

Accept-Encoding: gzip, deflate

Connection: keep-alive

Content-Length: 255

Content-type: application/xml

Host: sand-conf-server.my-example-server.com

User-Agent: HTTPie/0.9.9


<?xml version="1.0" encoding="UTF-8"?>
<SANDMessage xmlns="urn:mpeg:dash:schema:sandmessage:2016" senderId="abc1234"
generationTime="2016-02-21T11:20:52-08:00">

  <TcpList messageId="1234">

    <TcpConnection tcpid="socket1"/>

  </TcpList>

</SANDMessage>


HTTP/1.1 400 BAD REQUEST

Connection: keep-alive

Content-Length: 12

Content-Type: text/html; charset=utf-8

Date: Tue, 18 Jul 2017 12:38:20 GMT

Server: gunicorn/19.6.0

Via: 1.1 vegur


[RESULT][KO]
```

Server log

```
INFO:root:[TEST][KO] Content-Type (application/xml != application/sand+xml)
Element '{urn:mpeg:dash:schema:sandmessage:2016}TcpConnection', attribute 'tcpid':
'socket1' is not a valid value of the atomic type 'xs:unsignedInt'., line 4
INFO:root:[TEST][OK] HTTP method (PUT != POST)
INFO:root:[RESULT][KO]
INFO:root:[TEST][KO] SAND message validation
```

11.2.2.3 SAND WebSocket conformance server

A complete documentation is provided in the repository with dependencies and installation procedure. Basic usage information is provided here.

```
python sand_server.py run --port tcp_port
```

Example of valids request

```
INFO:root:[TEST][OK] Data frame type. Text frame received.


INFO:root:[TEST][OK] SAND message validation
```

Example of invalid request

```
INFO:root:[TEST][OK] Data frame type. Text frame received.
Element '{urn:mpeg:dash:schema:sandmessage:2016}BufferLevelList': Missing child
element(s). Expected is ( {urn:mpeg:dash:schema:sandmessage:2016}BufferLevel )., line 3
INFO:root:[TEST][KO] SAND message validation
```

11.3 Test vectors

Test vectors cover metrics, per, status messages and MPD with SAND signalling. Both valid and invalid test vectors are provided.

The test vectors are available at https://standards.iso.org/iso-iec/23009/-2/ed-3/en.

12 Conformance for ISO/IEC 23009-6 server push

12.1 Architecture

The conformance for ISO/IEC 23009-6 comprises a software module called PushDirective Validator written in the Python®[2] language. This module takes as input one or more input push directives or push acknowledgements. Each input is validated against the ABNF grammars defined in ISO/IEC 23009-6. The conformance module simply returns the number of correctly parsed valid examples and the number of incorrect inputs detected (see Figure 5).

2) Python is the trade name of a product supplied by The Python Software Foundation. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named.

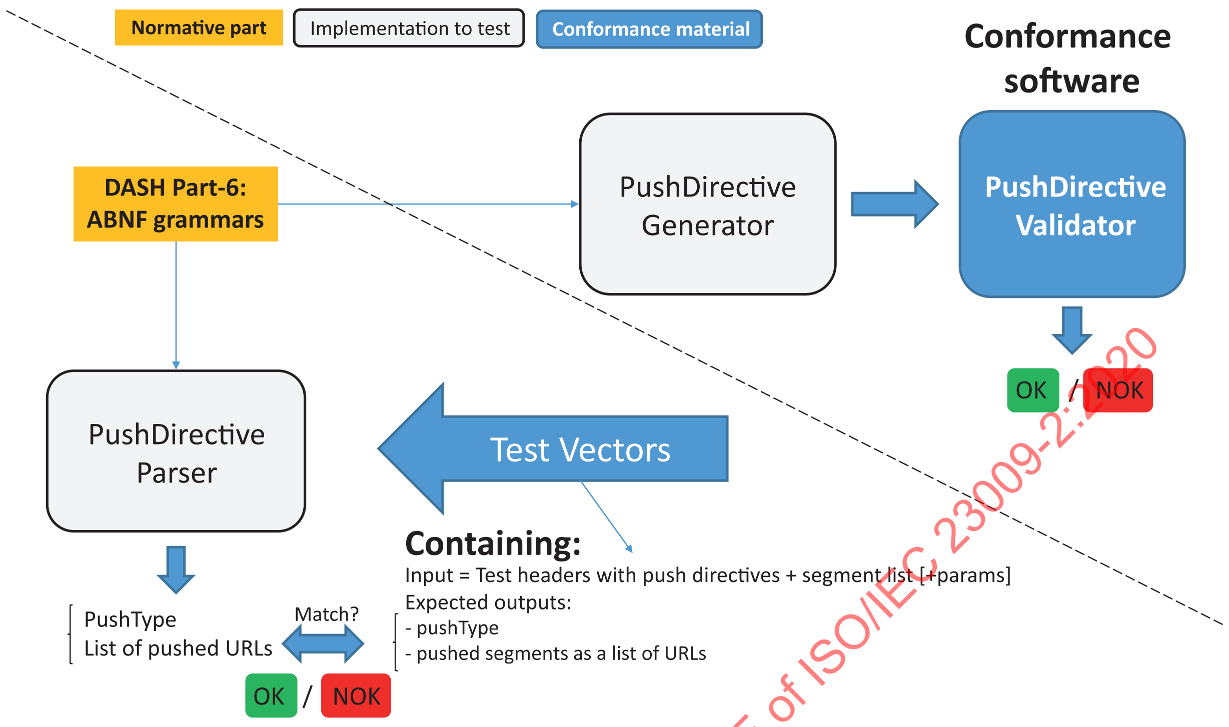

Figure 5 — Description of ISO/IEC 23009-6 conformance software

The Part-6 validator is written in Python language.

12.2 Status

From the list of defined PushType in ISO/IEC 23009-6:2017, Table 4, Table 11 indicates the level of support.

Table 11 — Level of support for the different push types defined in ISO/IEC 23009-6

PushType	Status
`urn:mpeg:dash:serverpush:2017:push-fast-start`	covered
`urn:mpeg:dash:serverpush:2017:push-list`	covered
`urn:mpeg:dash:serverpush:2017:push-next`	covered
`urn:mpeg:dash:serverpush:2017:push-none`	covered
`urn:mpeg:dash:serverpush:2017:push-template`	covered
`urn:mpeg:dash:serverpush:2017:push-time`	covered

For the push types accepting parameters defined by an ABNF grammar, indicates the level of support for each of these parameters.

For FastStartParams, the level of support is provided in Table 12.

Table 12 — Level of support for the FastStartParams defined in ISO/IEC 23009-6

Attribute Type	ABNF	Status
Initialization segments only	`ATTRIBUTE_ITEM = "init-only"`	covered

Media Type	`ATTRIBUTE_ITEM = "type=" MEDIATYPE`
	`MEDIATYPE = "video" / "audio"`
		covered

Start bitrate	`ATTRIBUTE_ITEM = "bitrate=" RATE`
	`RATE = SQUOTE INTEGER SQUOTE`
		covered

Resolution	`ATTRIBUTE_ITEM = "height=" RESOLUTION`
	`RESOLUTION = SQUOTE INTEGER SQUOTE`
		covered

Language	`ATTRIBUTE_ITEM = "lang=" SQUOTE STRING SQUOTE`	covered [a]

Media amount	`ATTRIBUTE_ITEM = "D=" SQUOTE INTEGER SQUOTE /`
	`"B=" SQUOTE INTEGER SQUOTE`
		covered

Media starting point	`ATTRIBUTE_ITEM = "t=" START_POINT`
	`START_POINT = "begin" / "now"`
		covered

URL List	`ATTRIBUTE_ITEM = "urls=[" URL_LIST "]"`	covered
Combination of the above parameters	`ATTRIBUTE_LIST = ATTRIBUTE_ITEM OWS ";" OWS ATTRIBUTE_LIST / ATTRIBUTE_ITEM`	covered

[a] Some exceptional cases may not be well supported (extended language subtags with their primary language).

This software is updated to support all the push types from ISO/IEC 23009-6:2017, Table 4.

12.3 Logistics

This conformance software is provided at https://standards.iso.org/iso-iec/23009/-2/ed-2. The zip file contains a readme file plus the following Python files:

1) DashPart6ABNF.py is the main file. Running this file launches the validation on two sets of push directives: one valid set and one invalid set. For the valid set, 100 % success is expected and for invalid test cases, the test is considered successful when 100 % expected failures is obtained.

2) DashPart6Lang.py contains parsing code for lang attribute used in push-fast-start. (This file also contains test samples from RFC 5646).

3) DashPart6RFC5646.py contains declaration related to RFC 5646 (for language).

4) DashPushFastStart.py contains parsing code for fast start pushDirective plus valid/invalid test samples.

5) DashPushList.py contains parsing cove for URL list push directive plus valid/invalid test samples.

6) DashPushNext.py contains the code for push-next directive validation plus valid/invalid test samples.

7) DashPushNone.py contains the code for push-none directive validation plus valid/invalid test samples.

8) DashPushTime.py contains the code for push-time directive validation plus valid/invalid test samples.

9) DashPushTypes.py contains definitions of DASH Part-6 types and grammars.

10) simpleTypeParser.py contains parsing routines for simple types like integer, URL and URL list.

11) urlparse_.py contains code for URL parsing.

To run successfully, this conformance software requires pyParsing module to be installed. It has been tested with Python interpreter version 3.6. under Windows®[3] 7 platform.

12.4 Usage

```
Usage: DASHPart6ABNF.py [-h] [--mode {0,1,2}]
                        [--level {debug,info,warning,error,critical}]
                        [--directive DIRECTIVE]

optional arguments:
  -h, --help            show this help message and exit
  --mode {0,1,2}, -m {0,1,2}
                        Kind of tests to run: valid(0) (the default);
                        invalid(1) or both(2)
  --level {debug,info,warning,error,critical}, -l {debug,info,warning,error,critical}
                        The logging level (as in python Logger, default =
                        "warning")
  --directive DIRECTIVE, -d DIRECTIVE
                        Tests a user-provided push-directive (as a string
                        between double-quotes). For example: -d
                        "\"urn:mpeg:dash:serverpush:2017:push-fast-
                        start\";init-only;"
```

There are three testing modes:

— running the grammar validator on valid test samples –m 0 (the default mode) or –m 2 to check that valid push directives or pushAck(s) are correctly parsed,

— running the grammar validator on invalid test samples –m 1 or –m 2 to check that expected failures are correctly detected,

— running the grammar validator against a user-provided push-directive. The software prints out whether the push-directive is valid or not.

For any testing mode, the level of logs can be controlled with the –l option (see Python Logger for the description of possible levels).

3) Windows is the trade name of a product supplied by Microsoft Corporation. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named.

Annex A
(normative)

MPD conformance checking

A.1 General

This annex provides detailed means for MPD conformance checking using the three steps outlined in Clause 5. Detailed examples and expected output behaviour are described for each step. The corresponding software and more examples are provided at https://standards.iso.org/iso-iec/23009/-2/ed-3/en.

A.2 Step 1: XLink resolver

A.2.1 General

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

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

b) 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.

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

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

e) If a URI reference is relative then reference resolution as defined in ISO/IEC 23009-1:2019, 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:

a) 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.

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

c) All XLink attributes shall be removed after dereferencing is completed.

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

A.2.2 Example 1

`ex01_xlink_valid.mpd` shows a valid MPD with XLink references. The output of the program is a success message for each step. This example comprises the following files: `ex01_xlink_valid.mpd`, `ex01_include.mpd`, and `ex01_include_1.mpd`.

A.2.3 Example 2

`ex02_xlink_invalid_circular.mpd` shows an invalid MPD with circular XLink references The program indicates the circular references with the following error message:

```
XLinkException: Circular referencing detected!
```

This example comprises the following files: `ex02_xlink_invalid_circular.mpd`, `ex02_include.mpd`, and `ex02_circular.mpd`.

A.2.4 Example 3

`ex03_xlink_invalid_ftp`.mpd shows an invalid MPD with XLink references with a wrong protocol. In this example, the FTP protocol is used instead of HTTP. The program indicates the wrong protocol with the following error message:

```
XLinkException: Only HTTP links are allowed!
```

This example comprises the following files: `ex03_xlink_invalid_ftp`.mpd.

A.2.5 Example 4

`ex04_xlink_invalid_wrongelement.mpd` shows an invalid MPD with XLink references where the XLink definition points to a different remote element than the local element. In this example, instead of pointing to a Period element, the XLink points to a Representation element. The program indicates the wrong element type with the following error message:

```
XLinkException: Referenced Document must contain same element type as referencing element!

Referencing element: Period
Referenced element: Representation
```

This example comprises the following files: `ex04_xlink_invalid_wrongelement.mpd` and `ex04_include.mpd`.

A.3 Step 2: XML validator

A.3.1 General

The XML Validator processes an MPD, checks that is it is well-formed, and validates it against the MPD schema defined in ISO/IEC 23009-1.

A.3.2 Example 1

`ex01_validator_valid.mpd` shows a valid MPD. The output of the program is a success message for each step.

A.3.3 Example 2

`ex02_validator_missingattribute.mpd` shows an invalid MPD, with a missing required attribute. The error occurs in the Representation element and is indicated with the following error message:

```
Line:Col[13:38]:cvc-complex-type.4: Attribute 'id' must appear on element
'Representation'.
XML validation not successful - DASH is not valid!
```

A.3.4 Example 3

`ex03_validator_wrongelement.mpd` shows an invalid MPD with an element at a wrong position. The error occurs after the Period element and is indicated by the following error message:

```
Line:Col[13:35]:cvc-complex-type.2.4.a: Invalid content was found
starting with element 'Representation'. One of
'{"urn:mpeg:dash:schema:mpd:2011":BaseURL,
"urn:mpeg:dash:schema:mpd:2011":SegmentBase,
"urn:mpeg:dash:schema:mpd:2011":SegmentList,
"urn:mpeg:dash:schema:mpd:2011":SegmentTemplate,
"urn:mpeg:dash:schema:mpd:2011":AdaptationSet,
"urn:mpeg:dash:schema:mpd:2011":Subset, WC[##other:
"urn:mpeg:dash:schema:mpd:2011"]}' is expected.
XML validation not successful - DASH is not valid!
```

A.3.5 Example 4

ex04_validator_missingprofile.mpd shows an invalid MPD with a missing profile attribute. The error occurs after the MPD element and is indicated by the following error message:

```
Line:Col[8:66]:cvc-complex-type.4: Attribute 'profiles' must appear on element 'MPD'.
XML validation not successful - DASH is not valid!
```

A.4 Step 3: Schematron validator

A.4.1 General

This subclause defines step 3 of the MPD conformance checking based on ISO/IEC 19757-3. The validation schema can be found in A.4.2. A description thereof (i.e. a more readable version of the rules and assertion messages) is provided in A.4.3. The MPDs for conformance checking are defined in A.4. The actual MPDs and corresponding code can be found attached to this document.

A.4.2 Schematron validation schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron" xmlns:dash="urn:mpeg:dash:schema:
mpd:2011" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xlink="http://www.
w3.org/1999/xlink" queryBinding='xslt' schemaVersion='ISO19757-3'>
   <ns prefix="dash" uri="urn:mpeg:dash:schema:mpd:2011"/>
   <ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>
   <ns prefix="xsi" uri="http://www.w3.org/2001/XMLSchema-instance"/>
   <title>Schema for validating MPDs</title>
   <pattern name="MPD element">
      <!-- R1.*: Check the conformance of MPD -->
      <rule context="dash:MPD">
         <!-- R1.0 -->
         <assert test="if (@type = 'dynamic' and not(@availabilityStartTime)) then false()
else true()">If MPD is of type "dynamic" availabilityStartTime shall be defined.</assert>
         <!-- R1.1 -->
         <assert test="if (@type = 'dynamic' and not(@publishTime)) then false() else
true()">If MPD is of type "dynamic" publishTime shall be defined.</assert>

         <!-- R1.3 -->
         <!-- <assert test="if (@type = 'static' and not(@mediaPresentationDuration)) then
false() else true()">If MPD is of type "static" mediaPresentationDuration shall be
defined.</assert>  -->
         <!-- R1.4 -->
         <assert test="if (@type = 'static' and descendant::dash:Period[1]/@start and
(years-from-duration(descendant::dash:Period[1]/@start) + months-from-
duration(descendant::dash:Period[1]/@start) + days-from-
duration(descendant::dash:Period[1]/@start) + hours-from-
duration(descendant::dash:Period[1]/@start) + minutes-from-
duration(descendant::dash:Period[1]/@start) +  seconds-from-
duration(descendant::dash:Period[1]/@start)) > 0) then false() else true()">If MPD is of
type "static" and the first period has a start attribute the start attribute shall be
zero.</assert>
         <!-- R1.5 -->
         <assert test="if (not(@mediaPresentationDuration) and not(@minimumUpdatePeriod))
then false() else true()">If mediaPresentationDuration is not defined for the MPD
minimumUpdatePeriod shall be defined or vice versa.</assert>
```

```
         <!-- R1.7 -->
         <assert test="if (not(@profiles) or (contains(@profiles,
'urn:mpeg:dash:profile:isoff-on-demand:2011') or contains(@profiles,
'urn:mpeg:dash:profile:isoff-live:2011') or contains(@profiles,
'urn:mpeg:dash:profile:isoff-main:2011') or contains(@profiles, 'urn:mpeg:dash:profile:f
ull:2011') or contains(@profiles, 'urn:mpeg:dash:profile:mp2t-main:2011') or contains(@
profiles, 'urn:mpeg:dash:profile:mp2t-simple:2011'))) then true() else false()">The
On-Demand profile shall be identified by the URN "urn:mpeg:dash:profile:isoff-on-
demand:2011". The live profile shall be identified by the URN
"urn:mpeg:dash:profile:isoff-live:2011". The main profile shall be identified by the URN
"urn:mpeg:dash:profile:isoff-main:2011". The full profile shall be identified by the URN
"urn:mpeg:dash:profile:full:2011". The mp2t-main profile shall be identified by the URN
"urn:mpeg:dash:profile:mp2t-main:2011". The mp2t-simple profile shall be identified by the
URN "urn:mpeg:dash:profile:mp2t-simple:2011".</assert>
         <!-- R1.8 -->
         <assert test="if (not(contains(@profiles, 'urn:mpeg:dash:profile:isoff-on-
demand:2011')) or not(@type) or @type='static') then true() else false()">For On-Demand
profile, the MPD @type shall be "static".</assert>
         <!-- R1.9 -->
         <assert test="if (not(@mediaPresentationDuration) and not(@minimumUpdatePeriod)
and not(dash:Period[last()]/@duration)) then false() else true()">If minimumUpdatePeriod
is not present and the last period does not include the duration attribute the
mediaPresentationDuration must be present.</assert>
         <!-- R1.10: Disabled, there is no such conformance point in DASH 2nd edition
(cuurent) -->
         <!-- assert test="if (@type='dynamic' and not(@id)) then false() else true()">If
the MPD type is dynamic, the id shall be present </assert-->

                </rule>
   </pattern>
   <pattern name="Period element">
      <!-- R2.*: Check the conformance of Period -->
      <rule context="dash:Period">
         <!-- R2.0 -->
         <assert test="if (string(@bitstreamSwitching) = 'true' and string(child::dash:Ada
ptationSet/@bitstreamSwitching) = 'false') then false() else true()">If bitstreamSwitching
is set to true all bitstreamSwitching declarations for AdaptationSet within this Period
shall not be set to false.</assert>
         <!-- R2.1 -->
         <assert test="if (@id = preceding::dash:Period/@id) then false() else true()">The
id of each Period shall be unique.</assert>
         <!-- R2.2: This rule has been found to not work properly, hence disabled for now
-->
         <!-- assert test="if ((years-from-duration(@start) + months-from-duration(@start)
+ days-from-duration(@start) + hours-from-duration(@start) + minutes-from-duration(@start)
+  seconds-from-duration(@start)) > (years-from-duration(following-sibling::dash:Period/@
start) + months-from-duration(following-sibling::dash:Period/@start) + days-from-
duration(following-sibling::dash:Period/@start) + hours-from-duration(following-
sibling::dash:Period/@start) + minutes-from-duration(following-sibling::dash:Period/@
start) +  seconds-from-duration(following-sibling::dash:Period/@start))) then false() else
true()">Periods shall be physically ordered in the MPD file in increasing order of their
start time.</assert-->
         <!-- R2.3 -->
         <assert test="if ((child::dash:SegmentBase and child::dash:SegmentTemplate and
child::dash:SegmentList) or (child::dash:SegmentBase and child::dash:SegmentTemplate) or
(child::dash:SegmentBase and child::dash:SegmentList) or (child::dash:SegmentTemplate and
child::dash:SegmentList)) then false() else true()">At most one of SegmentBase,
SegmentTemplate and SegmentList shall be defined in Period.</assert>
         <!-- R2.4 -->
         <assert test="if (not(@id) and ancestor::dash:MPD/@type = 'dynamic') then false()
else true()">If the MPD is dynamic the Period element shall have an id.</assert>
         <!-- R2.5 -->
         <assert test="if (not(descendant-or-self::dash:BaseURL) and not(descendant-or-
self::dash:SegmentTemplate) and not(descendant-or-self::dash:SegmentList) and not(@
xlink:href = 'urn:mpeg:dash:resolve-to-zero:2013')) then false() else true()">At least one
BaseURL, SegmentTemplate or SegmentList shall be defined in Period, AdaptationSet or
Representation.</assert>
               <assert test="if (@duration = 0 and count(child::dash:AdaptationSet)) then
false() else true()">If the duration attribute is set to zero, there should only be a
single AdaptationSet present.</assert>
```

```
                        <!--  DASH 8.3.2 -->
                        <assert test="if (contains(ancestor::dash:MPD/@profiles,
'urn:mpeg:dash:profile:isoff-on-demand:2011') and (child::dash:SegmentList or
child::dash:SegmentTemplate)) then false() else true()">Neither the Period.SegmentList
element nor the Period.SegmentTemplate element shall be present for On-Demand profile,
violated here. </assert>
      </rule>
   </pattern>
   <pattern name="AdaptationSet element">
      <!-- R3.*: Check the conformance of AdaptationSet -->
      <rule context="dash:AdaptationSet">
         <!-- R3.0 -->
         <assert test="if (@id = preceding-sibling::dash:AdaptationSet/@id) then false()
else true()">The id of each AdaptationSet within a Period shall be unique.</assert>

         <!-- R3.1 -->
         <assert test="if ((@lang = descendant::dash:ContentComponent/@lang) or (@
contentType = descendant::dash:ContentComponent/@contentType) or (@par = descendant::dash:
ContentComponent/@par)) then false() else true()">Attributes from the AdaptationSet shall
not be repeated in the descendanding ContentComponent elements.</assert>
         <!-- R3.2 -->
         <assert test="if ((@profiles and descendant::dash:Representation/@profiles) or (@
width and descendant::dash:Representation/@width) or (@height and descendant::dash:Represe
ntation/@height) or (@sar and descendant::dash:Representation/@sar) or (@frameRate and des
cendant::dash:Representation/@frameRate) or (@audioSamplingRate and descendant::dash:Repre
sentation/@audioSamplingRate) or (@mimeType and descendant::dash:Representation/@mimeType)
or (@segmentProfiles and descendant::dash:Representation/@segmentProfiles) or (@codecs and
descendant::dash:Representation/@codecs) or (@maximumSAPPeriod and descendant::dash:Repres
entation/@maximumSAPPeriod) or (@startWithSAP and descendant::dash:Representation/@
startWithSAP) or (@maxPlayoutRate and descendant::dash:Representation/@maxPlayoutRate) or
(@codingDependency and descendant::dash:Representation/@codingDependency) or (@scanType
and descendant::dash:Representation/@scanType)) then false() else true()">Common
attributes for AdaptationSet and Representation shall either be in one of the elements but
not in both.</assert>
         <!-- R3.3 -->
         <assert test="if ((@minWidth > @maxWidth) or (@minHeight > @maxHeight) or (@
minBandwidth > @maxBandwidth)) then false() else true()">Each minimum value (minWidth,
minHeight, minBandwidth) shall be larger than the maximum value.</assert>
         <!-- R3.4 -->
         <assert test="if (descendant::dash:Representation/@bandwidth &lt; @minBandwidth
or descendant::dash:Representation/@bandwidth > @maxBandwidth) then false() else
true()">The value of the bandwidth attribute shall be in the range defined by the
AdaptationSet.</assert>
         <!-- R3.5 -->
         <assert test="if (descendant::dash:Representation/@width > @maxWidth) then
false() else true()">The value of the width attribute shall be in the range defined by the
AdaptationSet.</assert>
         <!-- R3.6 -->
         <assert test="if (descendant::dash:Representation/@height > @maxHeight) then
false() else true()">The value of the height attribute shall be in the range defined by
the AdaptationSet.</assert>
         <!-- R3.7 -->
         <assert test="if (count(child::dash:Representation)=0) then false() else
true()">An AdaptationSet shall have at least one Representation element.</assert>
         <!-- R3.8 -->
         <assert test="if ((child::dash:SegmentBase and child::dash:SegmentTemplate and
child::dash:SegmentList) or (child::dash:SegmentBase and child::dash:SegmentTemplate) or
(child::dash:SegmentBase and child::dash:SegmentList) or (child::dash:SegmentTemplate and
child::dash:SegmentList)) then false() else true()">At most one of SegmentBase,
SegmentTemplate and SegmentList shall be defined in AdaptationSet.</assert>
         <!-- R3.9 -->
         <assert test="if ((@minFrameRate and (descendant::dash:Representation/@frameRate
&lt; @minFrameRate)) or (@maxFrameRate and (descendant::dash:Representation/@frameRate > @
maxFrameRate))) then false() else true()">ISO/IEC 23009-1:2019, 5.3.3.2: The value of the
frameRate attribute shall be in the range defined by the AdaptationSet.</assert>

      </rule>
   </pattern>

   <pattern name="ContentComponent element">
      <!-- R4.*: Check the conformance of ContentComponent -->
      <rule context="dash:ContentComponent">
         <!-- R4.0 -->
         <assert test="if (@id = preceding-sibling::dash:ContentComponent/@id) then
false() else true()">The id of each ContentComponent within an AdaptationSet shall be
unique.</assert>
      </rule>
   </pattern>
   <pattern name="Representation element">
      <!-- R5.*: Check the conformance of Representation -->
      <rule context="dash:Representation">
         <!-- R5.0 -->
         <assert test="if (not(@mimeType) and not(parent::dash:AdaptationSet/@mimeType))
then false() else true()">Either the Representation or the containing AdaptationSet shall
have the mimeType attribute.</assert>

         <!-- R5.1 -->
         <assert test="if (not(child::dash:SegmentTemplate or parent::dash:AdaptationSet/
dash:SegmentTemplate or ancestor::dash:Period/dash:SegmentTemplate) and (contains(@
profiles, 'urn:mpeg:dash:profile:isoff-live:2011') or contains(ancestor::dash:MPD/@
profiles, 'urn:mpeg:dash:profile:isoff-live:2011') or contains(parent::dash:Adaptation
Set/@profiles, 'urn:mpeg:dash:profile:isoff-live:2011'))) then false() else true()">For
live profile, 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.</assert>
         <!-- R5.2 -->
         <assert test="if ((child::dash:SegmentBase and child::dash:SegmentTemplate and
child::dash:SegmentList) or (child::dash:SegmentBase and child::dash:SegmentTemplate) or
(child::dash:SegmentBase and child::dash:SegmentList) or (child::dash:SegmentTemplate and
child::dash:SegmentList)) then false() else true()">At most one of SegmentBase,
SegmentTemplate and SegmentList shall be defined in Representation.</assert>

                        <!-- Unique ID check -->
         <assert test="if ((@id = preceding-sibling::dash:Representation/@id) or (@
id=parent::dash:AdaptationSet/preceding-sibling::dash:AdaptationSet/dash:Representation/@
id))then false() else true()">The id of each Representation within a Period shall be
unique.</assert>

      </rule>
   </pattern>
   <pattern name="SubRepresentation element">
      <!-- R6.*: Check the conformance of SubRepresentation -->
      <rule context="dash:SubRepresentation">
         <!-- R6.0 -->
         <assert test="if (@level and not(@bandwidth)) then false() else true()">If the
level attribute is defined for a SubRepresentation also the bandwidth attribute shall be
defined.</assert>

      </rule>
   </pattern>
   <pattern name="SegmentTemplate element">
      <!-- R7.*: Check the conformance of SegmentTemplate -->
      <rule context="dash:SegmentTemplate">
         <!-- R7.0 -->
         <assert test="if (not(@duration) and not(child::dash:SegmentTimeline) and not(@
initialization) ) then false() else true()">If more than one Media Segment is present the
duration attribute or SegmentTimeline element shall be present.</assert>
         <!-- R7.1 -->
         <assert test="if (@duration and child::dash:SegmentTimeline) then false() else
true()">Either the duration attribute or SegmentTimeline element shall be present but not
both.</assert>
         <!-- R7.2 -->
         <assert test="if (not(@indexRange) and @indexRangeExact) then false() else
true()">If indexRange is not present indexRangeExact shall not be present.</assert>
         <!-- R7.3 -->
         <assert test="if (@initialization and (matches(@initialization, '\$Number(%.
[^\$]*)?\$') or matches(@initialization, '\$Time(%.[^\$]*)?\$'))) then false() else
true()">Neither $Number$ nor the $Time$ identifier shall be included in the initialization
attribute.</assert>
         <!-- R7.4 -->
         <assert test="if (@bitstreamSwitching and (matches(@bitstreamSwitching,
'\$Number(%.[^\$]*)?\$') or matches(@bitstreamSwitching, '\$Time(%.[^\$]*)?\$'))) then
```

```
false() else true()">Neither $Number$ nor the $Time$ identifier shall be included in the
bitstreamSwitching attribute.</assert>
         <!-- R7.5-->
         <assert test="if (matches(@media, '\$.[^\$]*\$')) then every $y in (for $x in
tokenize(@media, '\$(Bandwidth|Time|Number|RepresentationID)(%.[^\$]*)?\$') return
matches($x, '\$.[^\$]*\$')) satisfies $y eq false() else true()">Only identifiers such as
$Bandwidth$, $Time$, $RepresentationID$, or $Number$ shall be used.</assert>
         <!-- R7.6-->
         <assert test="if (matches(@media, '\$RepresentationID%.[^\$]*\$')) then false()
else true()">$RepresentationID$ shall not have a format tag.</assert>
      </rule>
   </pattern>
   <pattern name="SegmentList element">
      <!-- R8.*: Check the conformance of SegmentList -->
      <rule context="dash:SegmentList">
         <!-- R8.0 -->
         <assert test="if (not(@duration) and not(child::dash:SegmentTimeline)) then if
(count(child::dash:SegmentURL) > 1) then false() else true() else true()">If more than one
Media Segment is present the duration attribute or SegmentTimeline element shall be
present.</assert>
         <!-- R8.1 -->
         <assert test="if (@duration and child::dash:SegmentTimeline) then false() else
true()">Either the duration attribute or SegmentTimeline element shall be present but not
both.</assert>
         <!-- R8.2 -->
         <assert test="if (not(@indexRange) and @indexRangeExact) then false() else
true()">If indexRange is not present indexRangeExact shall not be present.</assert>
      </rule>
   </pattern>
   <pattern name="SegmentBase element">
      <!-- R9.*: Check the conformance of SegmentBase -->
      <rule context="dash:SegmentBase">
         <!-- R9.0 -->
         <assert test="if (not(@indexRange) and @indexRangeExact) then false() else
true()">If indexRange is not present indexRangeExact shall not be present.</assert>
            <!-- R9.1 -->
            <assert test="if (@timeShiftBufferDepth) then if (@timeShiftbuffer &lt;
dash:MPD/@timeShiftBufferDepth) then false() else true() else true()">The
timeShiftBufferDepth shall not be smaller than timeShiftBufferDepth specified in the MPD
element</assert>
      </rule>
   </pattern>
   <pattern name="SegmentTimeline element">
      <!-- R10.*: Check the conformance of SegmentTimeline -->
      <rule context="dash:SegmentTimeline">
         <!-- R10.0 -->
         <assert test="if ((if (ancestor::dash:*[1]/@timescale) then (child::dash:S/@d div
ancestor::dash:*[1]/@timescale) else child::dash:S/@d) > (years-from-
duration(ancestor::dash:MPD/@maxSegmentDuration) + months-from-
duration(ancestor::dash:MPD/@maxSegmentDuration) + days-from-duration(ancestor::dash:MPD/@
maxSegmentDuration) + hours-from-duration(ancestor::dash:MPD/@maxSegmentDuration) +
minutes-from-duration(ancestor::dash:MPD/@maxSegmentDuration) +  seconds-from-
duration(ancestor::dash:MPD/@maxSegmentDuration))) then false() else true()">The d
attribute of a SegmentTimeline shall not exceed the value give bei the MPD
maxSegmentDuration attribute.</assert>
      </rule>
   </pattern>
   <pattern name="ProgramInformation element">
      <!-- R11.*: Check the conformance of ProgramInformation -->
      <rule context="dash:ProgramInformation">
         <!-- R11.0 -->
         <assert test="if (count(parent::dash:MPD/dash:ProgramInformation) > 1 and not(@
lang)) then false() else true()">If more than one ProgramInformation element is given each
ProgramInformation element shall have a lang attribute.</assert>
      </rule>
   </pattern>
   <pattern name="ContentProtection element">
      <!-- R12.*: Check the conformance of ContentProtection -->
      <rule context="dash:ContentProtection">
         <!-- R12.0 -->
         <assert test="if ((@schemeIdUri = 'urn:mpeg:dash:mp4protection:2011') and
not(string-length(@value) = 4)) then false() else true()">The value of ContentProtection
shall be the 4CC contained in the Scheme Type Box</assert>
         <!-- R12.1 -->
         <assert test="if ((@schemeIdUri = 'urn:mpeg:dash:13818:1:CA_descriptor:2011') and
not(string-length(@value) = 4)) then false() else true()">The value of ContentProtection
shall be the 4-digit lower-case hexadecimal Representation.</assert>

      </rule>
   </pattern>
   <pattern name="Role element">
      <!-- R13.*: Check the conformance of Role -->
      <rule context="dash:Role">
         <!-- R13.0 -->
         <assert test="if ((@schemeIdUri = 'urn:mpeg:dash:role:2011') and not(@value =
'caption' or @value = 'subtitle' or @value = 'main' or @value = 'alternate' or @value =
'supplementary' or @value = 'commentary' or @value = 'dub')) then false else true()">The
value of Role (role) shall be caption, subtitle, main, alternate, supplementary,
commentary or dub.</assert>
         <!-- R13.1 -->
         <assert test="if ((@schemeIdUri = 'urn:mpeg:dash:stereoid:2011') and not(starts-
with(@value, 'l') or starts-with(@value, 'r'))) then false() else true()">The value of
Role (stereoid) shall start with 'l' or 'r'.</assert>
      </rule>
   </pattern>
   <pattern name="FramePacking element">
      <!-- R14.*: Check the conformance of FramePacking -->
      <rule context="dash:FramePacking">
         <!-- R14.0 -->
         <assert test="if ((@schemeIdUri = 'urn:mpeg:dash:14496:10:frame_packing_
arrangement_type:2011') and not(contains(parent::dash:AdaptationSet/@codecs, 'avc') or con
tains(parent::dash:AdaptationSet/@codecs, 'svc') or contains(parent::dash:AdaptationSet/@
codecs, 'mvc')) and not(contains(parent::dash:Representation/@codecs, 'avc') or contains(p
arent::dash:Representation/@codecs, 'svc') or contains(parent::dash:Representation/@
codecs, 'mvc'))) then false() else true()">The URI urn:mpeg:dash:14496:10:frame_packing_
arrangement_type:2011 is used for Adaptation Sets or Representations that contain a video
component that conforms to ISO/IEC 14496-10.</assert>
         <!-- R14.1 -->
         <assert test="if ((@schemeIdUri = 'urn:mpeg:dash:13818:1:stereo_video_format_
type:2011') and not(parent::dash:AdaptationSet/@mimeType = 'video/mp2t') and not(parent::d
ash:Representation/@mimeType = 'video/mp2t')) then false() else true()">The URI
urn:mpeg:dash:13818:1:stereo_video_format_type:2011 is used for Adaptation Sets or
Representations that contain a video component that conforms to ISO/IEC 13818-1.</assert>
         <!-- R14.2 -->
         <assert test="if (not(@schemeIdUri = 'urn:mpeg:dash:14496:10:frame_packing_
arrangement_type:2011') and not(@schemeIdUri = 'urn:mpeg:dash:13818:1:stereo_video_format_
type:2011')) then false() else true()">schemeIdUri for FramePacking descriptor shall be
urn:mpeg:dash:14496:10:frame_packing_arrangement_type:2011 or
urn:mpeg:dash:13818:1:stereo_video_format_type:2011.</assert>
         <!-- R14.3 -->
         <assert test="if (not(@value = '0' or @value = '1' or @value = '2' or @value =
'3' or @value = '4' or @value = '5' or @value = '6')) then false() else true()">The value
of FramePacking shall be 0 to 6 as defined in ISO/IEC 23091(all parts).</assert>
      </rule>
   </pattern>
   <pattern name="AudioChannelConfiguration element">
      <!-- R15.*: Check the conformance of AudioChannelConfiguration -->
      <rule context="dash:AudioChannelConfiguration">
         <!-- R15.0 -->
         <assert test="if ((@schemeIdUri = 'urn:mpeg:dash:outputChannelPositionList:2012')
and not(count(tokenize(@value, ' ')) > 1)) then false() else true()">If URI urn:mpeg:das
h:outputChannelPositionList:2012 is used the value attribute shall be a space-delimited
list as defined in ISO/IEC 23091(all parts).</assert>

      </rule>
   </pattern>

   <pattern name="EventStream element">
      <!-- R16.*: Check the conformance of SegmentList -->
      <rule context="dash:EventStream">
         <!-- R16.0 -->
         <assert test="if (@actuate and not(@href)) then false() else true()">If href is
```

```
not present actuate shall not be present.</assert>
         <!-- R16.1 -->
         <assert test="if (not(@schemeIdUri)) then false() else true()">schemeIdUri shall
be present.</assert>
      </rule>
   </pattern>

      <pattern name="Subset element">
        <rule context="dash:Subset">
            <!--R17.1-->
           <assert test="if (@id = preceding::dash:Subset/@id) then false() else
true()">The id of each Subset shall be unique.</assert>
        </rule>
    </pattern>

   <pattern name="UTCTiming element">
      <!-- R18.*: Check the conformance of UTCTiming -->
      <rule context="dash:UTCTiming">
            <!-- R18.1 -->
         <assert test="if ((@schemeIdUri = 'urn:mpeg:dash:utc:ntp:2014') or (@schemeIdUri
= 'urn:mpeg:dash:utc:sntp:2014') or (@schemeIdUri = 'urn:mpeg:dash:utc:http-head:2014') or
(@schemeIdUri = 'urn:mpeg:dash:utc:http-xsdate:2014') or (@schemeIdUri =
'urn:mpeg:dash:utc:http-iso:2014') or (@schemeIdUri = 'urn:mpeg:dash:utc:http-ntp:2014')
or (@schemeIdUri = 'urn:mpeg:dash:utc:direct:2014')) then true() else false()">@
schemeIdUri for UTCTiming is not one of the 7 different types specified.</assert>
      </rule>
   </pattern>

   <pattern name="SupplementalProperty element">
      <!-- R19.*: Check the conformance of SupplementalProperty -->
      <rule context="dash:SupplementalProperty">
            <!-- R19.1 -->
         <assert test="if((@schemeIdUri= 'urn:mpeg:dash:chaining:2016') and
not((count(tokenize(@value, ','))=1) or (count(tokenize(@value, ','))>1)) )then false()
else true()">If schemeIdUri urn:mpeg:dash:chaining:2016 is used, then value attribute
shall be composed of the comma separated parameters (no comma needed if only first
parameter is present). </assert>
        <!-- R19.2 -->
         <assert test="if(not(parent::dash:MPD) and (@schemeIdUri=
'urn:mpeg:dash:fallback:2016') )then false() else true()">MPD fallback chaining shall be
signaled by Supplemental Descriptor on MPD level with schemeIdUri
urn:mpeg:dash:fallback:2016. </assert>
        <!-- R19.3 -->
         <assert test="if((@schemeIdUri= 'urn:mpeg:dash:fallback:2016') and
not((count(tokenize(@value, ' '))=1) or (count(tokenize(@value, ' '))>1)) )then false()
else true()">If schemeIdUri urn:mpeg:dash:fallback:2016 is used, then value attribute
shall be composed of one URL or whitespace separated URLs. </assert>
      </rule>
   </pattern>
        </schema>
```

A.4.3 Description of schematron rule

Description for MPD:

Rule	*Description*
if (@type = 'dynamic' and not(@availabilityStartTime)) then false() else true()	If MPD is of type "dynamic", availabilityStartTime shall be defined.
if (@type = 'dynamic' and not(@publishTime)), then false() else true()	If MPD is of type "dynamic", publishTime shall be defined.
if (@type = 'static' and @timeShiftBufferDepth) then false() else true()	If MPD is of type "static", timeShiftBufferDepth shall not be defined.
if (@type = 'static' and not(@mediaPresentationDuration)) then false() else true()	If MPD is of type "static", mediaPresentationDuration shall be defined.

Rule	*Description*
if (@type = 'static' and descendant::dash:Period[1]/@start and (years-from-duration(descendant::dash:Period[1]/@start) + months-from-duration(descendant::dash:Period[1]/@start) + days-from-duration(descendant::dash:Period[1]/@start) + hours-from-duration(descendant::dash:Period[1]/@start) + minutes-from-duration(descendant::dash:Period[1]/@start) + seconds-from-duration(descendant::dash:Period[1]/@start)) > 0) then false() else true()	If MPD is of type "static" and the first period has a start attribute, the start attribute shall be zero.
if (not(@mediaPresentationDuration) and not(@minimumUpdatePeriod)), then false() else true()	If mediaPresentationDuration is not defined for the MPD, minimumUpdatePeriod shall be defined or vice versa.
if (@type = 'static' and @minimumUpdatePeriod) then false() else true()	If MPD is of type "static", minimumUpdatePeriod shall not be defined.
if (not(@profiles) or (contains(@profiles, 'urn:mpeg:dash:profile:isoff-on-demand:2011') or contains(@profiles, 'urn:mpeg:dash:profile:isoff-live:2011') or contains(@profiles, 'urn:mpeg:dash:profile:isoff-main:2011') or contains(@profiles, 'urn:mpeg:dash:profile:full:2011') or contains(@profiles, 'urn:mpeg:dash:profile:mp2t-main:2011'))) then true() else false()	The On-Demand profile shall be identified by the URN "urn:mpeg:dash:profile:isoff-on-demand:2011". The live profile shall be identified by the URN "urn:mpeg:dash:profile:isoff-live:2011". The main profile shall be identified by the URN "urn:mpeg:dash:profile:isoff-main:2011". The full profile shall be identified by the URN "urn:mpeg:dash:profile:full:2011". The mp2t-main profile shall be identified by the URN "urn:mpeg:dash:profile:mp2t-main:2011". The mp2t-simple profile shall be identified by the URN "urn:mpeg:dash:profile:mp2t-simple:2011".
if (not(contains(@profiles, 'urn:mpeg:dash:profile:isoff-on-demand:2011')) or not(@type) or @type = 'static') then true() else false()	For On-Demand profile, the MPD @type shall be "static".

Description for `Period`:

Rule	*Description*
if (string(@bitstreamSwitching) = 'true' and string(child::dash:AdaptationSet/@bitstreamSwitching) = 'false') then false() else true()	If bitstreamSwitching is set to true, all bitstreamSwitching declarations for AdaptationSet within this Period shall not be set to false.
if (@id = preceding::dash:Period/@id), then false() else true()	The id of each Period shall be unique.

Rule	*Description*
if ((years-from-duration(@start) + months-from-duration(@start) + days-from-duration(@start) + hours-from-duration(@start) + minutes-from-duration(@start) + seconds-from-duration(@start)) > (years-from-duration(following-sibling::dash:Period/@start) + months-from-duration(following-sibling::dash:Period/@start) + days-from-duration(following-sibling::dash:Period/@start) + hours-from-duration(following-sibling::dash:Period/@start) + minutes-from-duration(following-sibling::dash:Period/@start) + seconds-from-duration(following-sibling::dash:Period/@start))) then false() else true()	Periods shall be physically ordered in the MPD file in increasing order of their start time.
if ((child::dash:SegmentBase and child::dash:SegmentTemplate and child::dash:SegmentList) or (child::dash:SegmentBase and child::dash:SegmentTemplate) or (child::dash:SegmentBase and child::dash:SegmentList) or (child::dash:SegmentTemplate and child::dash:SegmentList)) then false() else true()	At most one of SegmentBase, SegmentTemplate and SegmentList shall be defined in Period.
if (not(@id) and ancestor::dash:MPD/@type = 'dynamic') then false() else true()	If the MPD is dynamic the Period element shall have an id.
if (not(descendant-or-self::dash:BaseURL) and not(descendant-or-self::dash:SegmentTemplate) and not(descendant-or-self::dash:SegmentList)) then false() else true()	At least one BaseURL, SegmentTemplate or SegmentList shall be defined in Period, AdaptationSet or Representation.

Description for `AdaptationSet`:

Rule	*Description*
if (@id = preceding-sibling::dash:AdaptationSet/@id) then false() else true()	The id of each AdaptationSet within a Period shall be unique.
if ((@lang = descendant::dash:ContentComponent/@lang) or (@contentType = descendant::dash: ContentComponent/@content Type) or (@par = descendant::dash:ContentComponent/@par)) then false() else true()	Attributes from the AdaptationSet shall not be repeated in the descendanding ContentComponent elements.

Rule	*Description*
if ((@profiles and descendant::dash:Representation/@profiles) or (@width and descendant::dash:Representation/@width) or (@height and descendant::dash:Representation/@height) or (@sar and descendant::dash:Representation/@sar) or (@frameRate and descendant::dash:Representation/@frameRate) or (@audioSamplingRate and descendant::dash:Representation/@audioSampling Rate) or (@mimeType and descendant::dash:Representation/@mimeType) or (@segmentProfiles and descendant::dash:Representation/@segment Profiles) or (@codecs and descendant::dash:Representation/@codecs) or (@maximumSAPPeriod and descendant::dash:Representation/@maximumSAP Period) or (@startWithSAP and descendant::dash:Representation/@startWithSAP) or (@maxPlayoutRate and descendant::dash:Representation/@maxPlayout Rate) or (@codingDependency and descendant::dash:Representation/@coding Dependency) or (@scanType and descendant::dash:Representation/@scanType)) then false() else true()	Common attributes for AdaptationSet and Representation shall either be in one of the elements but not in both.
if ((@minWidth > @maxWidth) or (@minHeight > @maxHeight) or (@minBandwidth > @maxBandwidth)) then false() else true()	Each minimum value (minWidth, minHeight, minBandwidth) shall not be larger than the maximum value.
if (descendant::dash:Representation/@bandwidth &lt; @minBandwidth or descendant::dash:Representation/@bandwidth > @maxBandwidth) then false() else true()	The value of the bandwidth attribute shall be in the range defined by the AdaptationSet.
if (descendant::dash:Representation/@width &lt; @minWidth or descendant::dash:Representation/@width > @maxWidth) then false() else true()	The value of the width attribute shall be in the range defined by the AdaptationSet.
if (descendant::dash:Representation/@height &lt; @minHeight or descendant::dash:Representation/@height > @maxHeight),then false() else true()	The value of the height attribute shall be in the range defined by the AdaptationSet.
if (count(child::dash:Representation)=0) then false() else true()	An AdaptationSet shall have at least one Representation element.
if ((child::dash:SegmentBase and child::dash:SegmentTemplate and child::dash:SegmentList) or (child::dash:SegmentBase and child::dash:SegmentTemplate) or (child::dash:SegmentBase and child::dash:SegmentList) or (child::dash:SegmentTemplate and child::dash:SegmentList)) then false() else true()	At most, one of SegmentBase, SegmentTemplate and SegmentList shall be defined in AdaptationSet.

Description for `ContentComponent`:

Rule	*Description*

if (@id = preceding-sibling::dash:ContentComponent/@id) then false()
 	
else true()
	The id of each ContentComponent within an AdaptationSet shall be unique.

Description for `Representation`:

Rule	*Description*
if (not(@mimeType) and not(parent::dash:AdaptationSet/@mimeType)) then false() else true()	Either the Representation or the containing AdaptationSet shall have the mimeType attribute.
if (not(child::dash:SegmentTemplate or parent::dash:AdaptationSet/ dash:SegmentTemplate or ancestor::dash:Period/dash:SegmentTemplate) and (contains(@profiles, 'urn:mpeg:dash:profile:isoff-live:2011') or contains(ancestor::dash:MPD/@profiles, 'urn:mpeg:dash:profile:isoff-live:2011') or contains(parent::dash:AdaptationSet/@profiles, 'urn:mpeg:dash:profile:isoff-live:2011'))) then false() else true()	For live profile, 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.
if ((child::dash:SegmentBase and child::dash:SegmentTemplate and child::dash:SegmentList) or (child::dash:SegmentBase and child::dash:SegmentTemplate) or (child::dash:SegmentBase and child::dash:SegmentList) or (child::dash:SegmentTemplate and child::dash:SegmentList)) then false() else true()	At most, one of SegmentBase, SegmentTemplate and SegmentList shall be defined in Representation.

Description for `SubRepresentation`:

Rule	*Description*
if (@level and not(@bandwidth)), then false() else true()	If the level attribute is defined for a SubRepresentation, also the bandwidth attribute shall be defined.

Description for `SegmentTemplate`:

Rule	*Description*
if (not(@duration) and not(child::dash:SegmentTimeline)) then false() else true()	If more than one Media Segment is present, the duration attribute or SegmentTimeline element shall be present.
if (@duration and child::dash:SegmentTimeline) then false() else true()	Either the duration attribute or SegmentTimeline element shall be present but not both.
if (not(@indexRange) and @indexRangeExact) then false() else true()	If indexRange is not present, indexRangeExact shall not be present.
if (@initialization and (matches(@initialization, '\$Number(%.[^\$]*)?\$') or matches(@initialization, '\$Time(%.[^\$]*)?\$'))) then false() else true()	Neither $Number$ nor the $Time$ identifier shall be included in the initialization attribute.

Rule	*Description*
if (@bitstreamSwitching and (matches(@bitstreamSwitching, '\$Number(%.[^\$]*)?\$') or matches(@bitstreamSwitching, '\$Time(%.[^\$]*)?\$'))) then false() else true()	Neither $Number$ nor the $Time$ identifier shall be included in the bitstreamSwitching attribute.
if (matches(@media, '\$.[^\$]*\$')) then every $y in (for $x in tokenize(@media, '\$(Bandwidth\|Time\|Number\|RepresentationID)(%.[^\$]*)?\$') return matches($x, '\$.[^\$]*\$')) satisfies $y eq false() else true()	Only identifiers such as $Bandwidth$, $Time$, $RepresentationID$, or $Number$ shall be used.
if (matches(@media, '\$RepresentationID%.[^\$]*\$')) then false() else true()	$RepresentationID$ shall not have a format tag.

Description for `SegmentList`:

Rule	*Description*
if (not(@duration) and not(child::dash:SegmentTimeline)), then if (count(child::dash:SegmentURL) > 1) then false() else true() else true()	If more than one Media Segment is present, the duration attribute or SegmentTimeline element shall be present.
if (@duration and child::dash:SegmentTimeline), then false() else true()	Either the duration attribute or SegmentTimeline element shall be present but not both.
if (not(@indexRange) and @indexRangeExact) then false() else true()	If indexRange is not present, indexRangeExact shall not be present.

Description for `SegmentBase`:

Rule	*Description*
if (not(@indexRange) and @indexRangeExact) then false() else true()	If indexRange is not present, indexRangeExact shall not be present.

Description for `SegmentTimeline`:

Rule	*Description*
if ((if (ancestor::dash:*[1]/@timescale), then (child::dash:S/@d div ancestor::dash:*[1]/@timescale) else child::dash:S/@d) > (years-from-duration(ancestor::dash:MPD/@maxSegment Duration) + months-from-duration(ancestor::dash:MPD/@maxSegment Duration) + days-from-duration(ancestor::dash:MPD/@maxSegment Duration) + hours-from-duration(ancestor::dash:MPD/@maxSegment Duration) + minutes-from-duration(ancestor::dash:MPD/@maxSegment Duration) + seconds-from-duration(ancestor::dash:MPD/@maxSegment Duration))) then false() else true()	The d attribute of a SegmentTimeline shall not exceed the value give by the MPD maxSegmentDuration attribute.

Description for `ProgramInformation`:

Rule	*Description*
If (count(parent::dash:MPD/dash:ProgramInformation) > 1 and not(@lang)), then false() else true()	If more than one ProgramInformation element is given, each ProgramInformation element shall have a lang attribute.

Description for `ContentProtection`:

Rule	*Description*
if ((@schemeIdUri = 'urn:mpeg:dash:mp4protection:2011') and not(string-length(@value) = 4)) then false() else true()	The value of ContentProtection shall be the 4CC contained in the Scheme Type Box.
if ((@schemeIdUri = 'urn:mpeg:dash:13818:1:CA_descriptor:2011') and not(string-length(@value) = 4)) then false() else true()	The value of ContentProtection shall be the 4-digit lower-case hexadecimal Representation.

Description for `Role`:

Rule	*Description*
if ((@schemeIdUri = 'urn:mpeg:dash:role:2011') and not(@value = 'caption' or @value = 'subtitle' or @value = 'main' or @value = 'alternate' or @value = 'supplementary' or @value = 'commentary' or @value = 'dub')), then false else true()	The value of Role (role) shall be caption, subtitle, main, alternate, supplementary, commentary or dub.
if ((@schemeIdUri = 'urn:mpeg:dash:stereoid:2011') and not(starts-with(@value, 'l') or starts-with(@value, 'r'))), then false() else true()	The value of Role (stereoid) shall start with 'l' or 'r'.

Description for `FramePacking`:

Rule	*Description*
if ((@schemeIdUri = 'urn:mpeg:dash:14496:10:frame_packing_arrangement_type:2011') and not(contains(parent::dash:AdaptationSet/@codecs, 'avc') or contains(parent::dash:AdaptationSet/@codecs, 'svc') or contains(parent::dash:AdaptationSet/@codecs, 'mvc')) and not(contains(parent::dash:Representation/@codecs, 'avc') or contains(parent::dash:Representation/@codecs, 'svc') or contains(parent::dash:Representation/@codecs, 'mvc'))) then false() else true()	The URI urn:mpeg:dash:14496:10:frame_packing_arrangement_type:2011 is used for Adaptation Sets or Representations that contain a video component that conforms to ISO/IEC 14496-10.
if ((@schemeIdUri = 'urn:mpeg:dash:13818:1:stereo_video_format_type:2011') and not(parent::dash:AdaptationSet/@mimeType = 'video/mp2t') and not(parent::dash:Representation/@mimeType = 'video/mp2t')) then false() else true()	The URI urn:mpeg:dash:13818:1:stereo_video_format_type:2011 is used for Adaptation Sets or Representations that contain a video component that conforms to ISO/IEC 13818-1.
if (not(@schemeIdUri = 'urn:mpeg:dash:14496:10:frame_packing_arrangement_type:2011') and not(@schemeIdUri = 'urn:mpeg:dash:13818:1:stereo_video_format_type:2011')), then false() else true()	schemeIdUri for FramePacking descriptor shall be urn:mpeg:dash:14496:10:frame_packing_arrangement_type:2011 or urn:mpeg:dash:13818:1:stereo_video_format_type:2011.
if (not(@value = '0' or @value = '1' or @value = '2' or @value = '3' or @value = '4' or @value = '5' or @value = '6')) then false() else true()	The value of FramePacking shall be 0 to 6 as defined in ISO/IEC 23091 (all parts).

Description for `AudioChannelConfiguration`:

Rule	*Description*
if ((@schemeIdUri = 'urn:mpeg:dash:outputChannelPositionList:2012') and not(count(tokenize(@value, ' ')) > 1)) then false() else true()	If URI urn:mpeg:dash:outputChannelPositionList:2012 is used the value attribute shall be a space-delimited list as defined in ISO/IEC 23091 (all parts).

Description for `Subset`:

Rule	*Description*
if (@id = preceding-sibling::dash:Subset/@id) then false() else true()	The id of each Subset within a Period shall be unique.

Description for `EventStream`:

Rule	*Description*
if ((@schemeIdUri = preceding::dash:EventStream/@schemeIdUri) and (@value = preceding::dash:EventStream/@value)) then false() else true()	A Period shall contain at most one EventStream element with the same value for the schemeIdUri attribute and value attribute.
if ((@schemeIdUri = 'urn:mpeg:dash:event:2012') and not(@value = '1' or @value = '2')) then false() else true()	If URI urn:mpeg:dash:event:2012 is used the value attribute shall have the value 1 or 2.

Description for `InbandEventStream`:

Rule	*Description*
if ((@schemeIdUri = preceding::dash:InbandEventStream/@schemeIdUri) and (@value = preceding::dash:InbandEventStream/@value)) then false() else true()	There shall be only one InbandEventStream element with the same value for the schemeIdUri attribute and value attribute.
if ((@schemeIdUri = 'urn:mpeg:dash:event:2012') and not(@value = '1' or @value = '2')) then false() else true()	If URI urn:mpeg:dash:event:2012 is used the value attribute shall have the value 1 or 2.

Description for `Event`:

Rule	*Description*
if (if (not(@presentationTime)), then (0 &lt; preceding-sibling::dash:Event/@presentationTime) else (@presentationTime &lt; preceding-sibling::dash:Event/@presentationTime)) then false() else true()	Events shall be in non-decreasing order based on the presentation time.

Description for `UTCTiming`:

Rule	*Description*
if ((@schemeIdUri = 'urn:mpeg:dash:utc:ntp:2014') or (@schemeIdUri = 'urn:mpeg:dash:utc:sntp:2014') or (@schemeIdUri = 'urn:mpeg:dash:utc:http-head:2014') or (@schemeIdUri = 'urn:mpeg:dash:utc:http-xsdate:2014') or (@schemeIdUri = 'urn:mpeg:dash:utc:http-iso:2014') or (@schemeIdUri = 'urn:mpeg:dash:utc:http-ntp:2014') or (@schemeIdUri = 'urn:mpeg:dash:utc:direct:2014')) then true() else false()	Ref: @schemeIdUri for UTCTiming Table X – Different UTC timing methods.

Description for `Representation`:

Rule	*Description*
if (not(@associationId) and @associationType) then false() else true()	The attribute(@associationType) shall not be present when @associationId is not present.
if (count(tokenize (@associationType, ' ')) != count(tokenize (@associationId))) then false() else true()	The attribute(@associationType) shall have as many values as the number of identifiers declared in the @associationId attribute.
if (@associationType and not(every $x in tokenize(@associationType,' ') satisfies string-length($x)= 4)) then false() else true()	The value of the attribute @associationType shall be the 4 characters codes defined in ISOBMFF fro the track reference type.
if (@associationType and not(every $x in tokenize(@associationType,' ') satisfies string-length($x)= 4)) then false() else true()	Representation(s) indicated in Representation@associationId is(are) not present

Description for Spatial Relationship Description:

Rule	*Description*
`if (every $AdaptationSet in child::dash:AdaptationSet satisfies $AdaptationSet/dash:EssentialProperty/@schemeIdUri = 'urn:mpeg:dash:srd:2014') then false() else true()`	When every Adaptation Set in an MPD has a SRD descriptor, at least one of these descriptors shall be a SupplementalProperty.
`if (@schemeIdUri = 'urn:mpeg:dash:srd:2014' and (parent::dash:AdaptationSet or parent::dash:SubRepresentation)) then true() else false()`	An EssentialProperty or SupplementalProperty descriptor with @schemeIdUri equal to "urn:mpeg:dash:srd:2014" shall be the child element of an AdaptationSet or a SubRepresentation element.
`if (@schemeIdUri = 'urn:mpeg:dash:srd:2014' and @value) then true() else false()`	If an EssentialProperty or SupplementalProperty descriptor with @schemeIdUri equal to "urn:mpeg:dash:srd:2014" is present, then the @value attribute must be present.
`if (@schemeIdUri = 'urn:mpeg:dash:srd:2014' and matches(@value, '^\d+,\d+,\d+,\d+,\d+')) then true() else false()`	If an EssentialProperty or SupplementalProperty descriptor with @schemeIdUri equal to "urn:mpeg:dash:srd:2014" is present, then the @value attribute must contain at least the mandatory comma separated parameters, i.e. `source_id, x, y, w, h`.
`if (@schemeIdUri = 'urn:mpeg:dash:srd:2014' and matches(@value, '^(\d+,){4,7}\d+$')) then true() else false()`	If an EssentialProperty or SupplementalProperty descriptor with @schemeIdUri equal to "urn:mpeg:dash:srd:2014" is present, then each parameter value has to match the expected type format i.e. non-negative integer in decimal representation.

Rule	*Description*
`if (@schemeIdUri = 'urn:mpeg:dash:srd:2014' and matches(@value, '^(\d+,){5}\d+$')) then false() else true()`	If an EssentialProperty or SupplementalProperty descriptor with @schemeIdUri equal to “urn:mpeg:dash:srd:2014” is present and the @value attribute contains the optional parameter `W` then the optional parameter `H` shall be present too.
	If an EssentialProperty or SupplementalProperty descriptor with @schemeIdUri equal to “urn:mpeg:dash:srd:2014” is present and the @value attribute contains the optional parameter `H` then the optional parameter `W` shall be present too.
	If an EssentialProperty or SupplementalProperty descriptor with @schemeIdUri equal to “urn:mpeg:dash:srd:2014” is present and the @value attribute contains the optional parameter `spatial_set_id` then the optional parameters `W` and `H` shall be present too.
`if (some $source_id in (for $srd in descendant::dash:*[@schemeIdUri = 'urn:mpeg:dash:srd:2014'] return subsequence(tokenize($srd/@value,','), 1,1)) satisfies (every $srd in descendant::dash:*[@schemeIdUri = 'urn:mpeg:dash:srd:2014' and subsequence(tokenize(@value,','),1,1) = $source_id] satisfies matches($srd/@value, '^(\d+,){4}\d+$'))) then false() else true()`	For a given `source_id` of the @value attribute, at least one of the EssentialProperty or SupplementalProperty in the containing Period shall specify the optional parameters `W` and `H`.
`if(count(descendant::dash:*[(@schemeIdUri = 'urn:mpeg:dash:srd:2014')]) > 0) then if (some $source_id in (for $srd in descendant::dash:*[@schemeIdUri = 'urn:mpeg:dash:srd:2014'] return subsequence(tokenize($srd/@value,','), 1,1)) satisfies (if (count(distinct-values(for $srd in descendant::dash:*[@schemeIdUri = 'urn:mpeg:dash:srd:2014' and subsequence(tokenize(@value,','),1,1) = $source_id and matches(@value, '^(\d+,){6}\d+')] return concat(string(subsequence(tokenize($srd/@value,','),6,1)), string(subsequence(tokenize($srd/@value,','),7,1))))) > 1) then every $srd in descendant::dash:*[@schemeIdUri = 'urn:mpeg:dash:srd:2014' and subsequence(tokenize(@value,','),1,1) = $source_id] satisfies matches($srd/@value, '^(\d+,){6}\d+') else true())) then true() else false() else true()`	For a given `source_id` of the @value attribute, if two SRD elements (indistinctively EssentialProperty or SupplementalProperty) explicitly specify a different pair of values for the optional parameters (`W`,`H`) then all the remaining SRD elements shall explicitly specify a pair of values for (`W`,`H`) too.

Rule	*Description*
W is present in the descriptor `every $srd in descendant::dash:*[@schemeIdUri = 'urn:mpeg:dash:srd:2014' and matches(@value, '^(\d+,){6}\d+')] satisfies sum((number(subsequence(tokenize($srd/@value,','),2,1)),number(subsequence(tokenize($srd/@value,','),4,1)) ) ) le number(subsequence(tokenize($srd/@value,','),6,1))`	For a given `source_id` of the @value attribute, the values of x, w and `W` shall be such that, for each descriptor, the sum of `x` and `w` is smaller than or equal to `W`.
W is not present in the descriptor `if(count(descendant::dash:*[(@schemeIdUri = 'urn:mpeg:dash:srd:2014')]) > 0) then if (some $source_id in (for $srd in descendant::dash:*[@schemeIdUri = 'urn:mpeg:dash:srd:2014'] return subsequence(tokenize($srd/@value,','), 1,1)) satisfies (every $srd in descendant::dash:*[@schemeIdUri = 'urn:mpeg:dash:srd:2014' and subsequence(tokenize(@value,','),1,1) = $source_id and matches(@value, '^(\d+,){4}\d+')] satisfies sum((number(subsequence(tokenize($srd/@value,','),2,1)),number(subsequence(tokenize($srd/@value,','),4,1)))) le number(subsequence(tokenize(descendant::dash:*[(@schemeIdUri = 'urn:mpeg:dash:srd:2014') and (subsequence(tokenize(@value,','),1,1) = $source_id) and (matches(@value, '^(\d+,){6}\d+'))]/@value,','),6,1)))) then true() else false() else true()`	
H is present in the descriptor `every $srd in descendant::dash:*[@schemeIdUri = 'urn:mpeg:dash:srd:2014' and matches(@value, '^(\d+,){6}\d+')] satisfies sum((number(subsequence(tokenize($srd/@value,','),3,1)),number(subsequence(tokenize($srd/@value,','),5,1)) ) ) le number(subsequence(tokenize($srd/@value,','),7,1))`	For a given `source_id` of the @value attribute, the values of y, h and `H` shall be such that, for each descriptor, the sum of `y` and `h` is smaller than or equal to `H`.

Rule	*Description*
H is not present in the descriptor `if(count(descendant::dash:*[(@scheme IdUri = 'urn:mpeg:dash:srd:2014')]) > 0) then if (some $source_id in (for $srd in descendant::dash:*[(@schemeIdUri = 'urn:mpeg:dash:srd:2014')] return subsequence(tokenize($srd/@value,',') ,1,1)) satisfies (every $srd in descendant::dash:*[(@schemeIdUri = 'urn:mpeg:dash:srd:2014') and (subsequence(tokenize(@value,','),1 ,1) = $source_id) and (matches(@value, '^(\d+,){4}\d+'))] satisfies sum((number(subsequence(tokenize ($srd/@value,','),3,1)),number (subsequence(tokenize($srd/@value, ','),5,1)))) le number(subsequence(tokenize (descendant::dash:*[(@schemeIdUri = 'urn:mpeg:dash:srd:2014') and subsequence(tokenize(@value,','),1 ,1) = $source_id and matches(@value, '^(\d+,){6}\d+')]/@value,','),7,1)))) then true() else false() else true()`	

A.4.4 MPD examples for conformance checking

A.4.4.1 General

The following commands are used to transform the validation schema.

NOTE The validation schema is called `schematron.xsd`.

```
java -jar saxon/saxon9.jar -versionmsg:off -s:schematron.xsd
     -o:tmp/new_schema1.sch -xsl:schematron/iso_dsdl_include.xsl
java -jar saxon/saxon9.jar -versionmsg:off -s:tmp/new_schema1.sch
     -o:tmp/new_schema2.sch -xsl:schematron/iso_abstract_expand.xsl
java -jar saxon/saxon9.jar -versionmsg:off -s:tmp/new_schema2.sch
     -o:output/val_schema.xsl -xsl:schematron/iso_svrl_for_xslt1.xsl
```

The first two commands resolve inclusions and abstractions. The last command generates the XSLT stylesheet, which can then be applied to the XML description. The command generates an XSLT 1.0 description.

The following MPD examples are split up as following:

— example XML;

— command to validate the example;

— output generated during the validation;

— description of erroneous output (relevant parts).

A.4.4.2 Example 1

`ex01.mpd` is a valid DASH MPD and the result is `result_ex01.xml`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex01.mpd
     -o:output/result_ex01.xml -xsl:output/val_schema.xsl
```

A.4.4.3 Example 2

`ex02.mpd` is an invalid DASH MPD and the result is `result_ex02.xml`. The error occurs in the MPD. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex02.mpd
     -o:output/result_ex02.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 19 to 22. Lines 19 and 20 show a failed assertion. The test which failed is also shown: “`if (@type = 'dynamic' and not(@availabilityStartTime)) then false() else true()`“. Line 21 shows the error message “`If MPD is of type "dynamic" availabilityStartTime shall be defined.`“

A.4.4.4 Example 3

`ex03.mpd` is an invalid DASH MPD and the result i `result_ex03.xml`. The error occurs in the MPD. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex03.mpd
     -o:output/result_ex03.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 19 to 22. Lines 19 and 20 define a failed assertion. The test which failed is also shown: “`if (@type = 'static' and @timeShiftBufferDepth) then false() else true()`“. Line 21 shows the error message: “`If MPD is of type "static" timeShiftBufferDepth shall not be defined.`“

A.4.4.5 Example 4

`ex04.mpd` is an invalid DASH MPD and the result is `result_ex04.xml`. There are two errors which occur in the MPD. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex04.mpd
     -o:output/result_ex04.xml -xsl:output/val_schema.xsl
```

The validation errors are shown in the lines 19 to 22 and lines 23 to 26. Lines 19 and 20 show the first failed assertion. The test which failed is also shown: “`if (@type = 'static' and not(@mediaPresentationDuration)) then false() else true()`”. Line 21 shows the error message: “`If MPD is of type "static" mediaPresentationDuration shall be defined.`” Furthermore, there is another failed assertion shown in lines 23 and 24: “`if (not(@mediaPresentationDuration) and not(@minimumUpdatePeriod)) then false() else true()`“. Line 25 shows the error message: “`If mediaPresentationDuration is not defined for the MPD minimumUpdatePeriod shall be defined or vice versa.`”

A.4.4.6 Example 5

`ex05.mpd` is an invalid DASH MPD and the result is `result_ex05.xml`. The error occurs in the MPD. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex05.mpd
     -o:output/result_ex05.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 19 to 22. Lines 19 and 20 show a failed assertion. The test which failed is also shown: “`if (@type = 'static' and descendant::dash:Period[1]/@start and (years-from-duration(descendant::dash:Period[1]/@start) + months-from-duration(descendant::dash:Period[]]/@start) + days-from-`

duration(descendant::dash:Period[1]/@start) + hours-from-duration(descendant::dash:Period[1]/@start) + minutes-from-duration(descendant::dash:Period[1]/@start) + seconds-from-duration(descendant::dash:Period[1]/@start)) &gt; 0) then false() else true()". Line 21 shows the error message: "If MPD is of type "static" and the first period has a start attribute the start attribute shall be zero."

A.4.4.7 Example 6

ex06.mpd is an invalid DASH MPD and the result is result_ex06.xml. There are two errors which occur in the MPD. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex06.mpd
     -o:output/result_ex06.xml -xsl:output/val_schema.xsl
```

The validation errors are shown in the lines 19 to 22 and lines 23 to 26. Lines 19 and 20 show the first failed assertion. The test which failed is also shown: "if (@type = 'static' and not(@mediaPresentationDuration)) then false() else true()". Line 21 shows the error message: "If MPD is of type "static" mediaPresentationDuration shall be defined." Furthermore, there is another failed assertion shown in lines 23 and 24: "if (@type = 'static' and @minimumUpdatePeriod) then false() else true()". Line 25 shows the error message: "If MPD is of type "static" minimumUpdatePeriod shall not be defined."

A.4.4.8 Example 7

ex07.mpd is an invalid DASH MPD and the result is result_ex07.xml. The error occurs in the Period. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex07.mpd
     -o:output/result_ex07.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 21 to 24. Lines 21 and 22 show a failed assertion. The test which failed is also shown: "if (string(@bitstreamSwitching) = 'true' and string(child::dash:AdaptationSet/@bitstreamSwitching) = 'false') then false() else true()". Line 23 shows the error message: "If bitstreamSwitching is set to true all bitstreamSwitching declarations for AdaptationSet within this Period shall not be set to false."

A.4.4.9 Example 8

ex08.mpd is an invalid DASH MPD and the result is result_ex08.xml. The error occurs in the Period. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex08.mpd
     -o:output/result_ex08.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 22 to 25. Lines 22 and 23 show a failed assertion. The test which failed is also shown: "if ((@id = preceding::dash:Period/@id)) then false() else true()". Line 24 shows the error message: "The id of each Period shall be unique."

A.4.4.10 Example 9

ex09.mpd is an invalid DASH MPD and the result is result_ex09.xml. The error occurs in the Period. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex09.mpd
     -o:output/result_ex09.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 22 to 25. Lines 22 and 23 show a failed assertion. The test which failed is also shown: "if ((years-from-duration(@start) + months-from-duration(@start) + days-from-duration(@start) + hours-from-duration(@start) + minutes-from-duration(@start) + seconds-from-duration(@start)) &gt; (years-from-duration(following-sibling::dash:Period/@start) + months-from-duration(following-sibling::dash:Period/@start) + days-from-duration(following-sibling::dash:Period/@start) + hours-from-duration(following-sibling::dash:Period/@start) + minutes-from-duration(following-sibling::dash:Period/@start) + seconds-from-duration(following-sibling::dash:Period/@start))) then false() else true()". Line 24 shows the error message: "Periods shall be physically ordered in the MPD file in increasing order of their start time."

A.4.4.11 Example 10

ex10.mpd is an invalid DASH MPD and the result is result_ex10.xml. The error occurs in the AdaptationSet. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex10.mpd
     -o:output/result_ex10.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 24 to 27. Lines 24 and 25 show a failed assertion. The test which failed is also shown: "if (@id = preceding-sibling::dash:AdaptationSet/@id) then false() else true()". Line 26 shows the error message: "The id of each AdaptationSet within a Period shall be unique."

A.4.4.12 Example 11

ex11.mpd is an invalid DASH MPD and the result is result_ex11.xml. The error occurs in the AdaptationSet. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex11.mpd
     -o:output/result_ex11.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 23 to 26. Lines 23 and 24 show a failed assertion. The test which failed is also shown: "if ((@lang = descendant::dash:ContentComponent/@lang) or (@contentType = descendant::dash:ContentComponent/@contentType) or (@par = descendant::dash:ContentComponent/@par)) then false() else true()". Line 25 shows the error message: "Attributes from the AdaptationSet shall not be repeated in the descendanding ContentComponent elements."

A.4.4.13 Example 12

ex12.mpd is an invalid DASH MPD and the result is result_ex12.xml. The error occurs in the AdaptationSet. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex12.mpd
     -o:output/result_ex12.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 23 to 26. Lines 23 and 24 show a failed assertion. The test which
failed is also shown: "if ((@profiles and
descendant::dash:Representation/@profiles) or (@width and
descendant::dash:Representation/@width) or (@height and
descendant::dash:Representation/@height) or (@sar and
descendant::dash:Representation/@sar) or (@frameRate and
descendant::dash:Representation/@frameRate) or (@audioSamplingRate and
descendant::dash:Representation/@audioSamplingRate) or (@mimeType and
descendant::dash:Representation/@mimeType) or (@segmentProfiles and
descendant::dash:Representation/@segmentProfiles) or (@codecs and
descendant::dash:Representation/@codecs) or (@maximumSAPPeriod and
descendant::dash:Representation/@maximumSAPPeriod) or (@startWithSAP and
descendant::dash:Representation/@startWithSAP) or (@maxPlayoutRate and

descendant::dash:Representation/@maxPlayoutRate) or (@codingDependency and descendant::dash:Representation/@codingDependency) or (@scanType and descendant::dash:Representation/@scanType)) then false() else true()". Line 25 shows the error message: "Common attributes for AdaptationSet and Representation shall either be in one of the elements but not in both."

A.4.4.14 Example 13

ex13.mpd is an invalid DASH MPD and the result is result_ex13.xml. There are two errors occurring in AdaptationSet. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex13.mpd
     -o:output/result_ex13.xml -xsl:output/val_schema.xsl
```

The validation errors are shown in the lines 23 to 26 and lines 27 to 30. Lines 23 and 24 show the first failed assertion. The test which failed is also shown: "if ((@minWidth &gt; @maxWidth) or (@minHeight &gt; @maxHeight) or (@minBandwidth &gt; @maxBandwidth)) then false() else true()". Line 25 shows the error message: "Each minimum value (minWidth, minHeight, minBandwidth) shall be larger than the maximum value." Furthermore, there is another failed assertion shown in lines 27 and 28: "if (descendant::dash:Representation/@bandwidth &lt; @minBandwidth or descendant::dash:Representation/@bandwidth &gt; @maxBandwidth) then false() else true()". Line 29 shows the error message: "The value of the bandwidth attribute shall be in the range defined by the AdaptationSet."

A.4.4.15 Example 14

ex14.mpd is an invalid DASH MPD and the result is in result_ex14.xml. There are two errors occurring in AdaptationSet. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex14.mpd
     -o:output/result_ex14.xml -xsl:output/val_schema.xsl
```

The validation errors are shown in the lines 23 to 26 and lines 27 to 30. Lines 23 and 24 show the first failed assertion. The test which failed is also shown: "if (descendant::dash:Representation/@width &lt; @minWidth or descendant::dash:Representation/@width &gt; @maxWidth) then false() else true()". Line 25 shows the error message: "The value of the width attribute shall be in the range defined by the AdaptationSet." Lines 27 and 28 show the second failed assertion. The second test which failed is also shown: "if (descendant::dash:Representation/@height &lt; @minHeight or descendant::dash:Representation/@height &gt; @maxHeight) then false() else true()". Line 29 shows the error message: "The value of the height attribute shall be in the range defined by the AdaptationSet."

A.4.4.16 Example 15

ex15.mpd is an invalid DASH MPD and the result is result_ex15.xml. There are two errors occuring, one in the Period and one in the AdaptationSet. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex15.mpd
     -o:output/result_ex15.xml -xsl:output/val_schema.xsl
```

The validation are shown in lines 21 to 24 and lines 27 to 30. Lines 21 and 22 show the first failed assertion. The test which failed is also shown: "if (not(descendant-or-self::dash:BaseURL) and not(descendant-or-self::dash:SegmentTemplate) and not(descendant-or-self::dash:SegmentList)) then false() else true()". Line 23 shows the error message: "At least one BaseURL, SegmentTemplate or SegmentList shall be defined in Period, AdaptationSet or Representation." Lines 27 and 28 show the second failed assertion. The second test which failed is also shown: "if (count(child::dash:Representation)=0) then false() else true()". Line 29 shows the error message: "An AdaptationSet shall have at least one Representation element."

A.4.4.17 Example 16

`ex16.mpd` is an invalid DASH MPD and the result is `result_ex16.xml`. The error occurs in the `ContentComponent`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex16.mpd
     -o:output/result_ex16.xml -xsl:output/val_schema.xsl
```

The validation error is shown in lines 27 to 30. Lines 27 and 28 show a failed assertion. The test which failed is also shown: "`if (@id = preceding-sibling::dash:ContentComponent/@id) then false() else true()`". Line 29 shows the error message: "`The id of each ContentComponent within an AdaptationSet shall be unique.`"

A.4.4.18 Example 17

`ex17.mpd` shows an invalid DASH MPD with the result in `result_ex17.xml`. There is one error occurring in Representation. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex17.mpd
     -o:output/result_ex17.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 26 to 29. .Lines 26 and 27 defined the first failed assertion. The test which failed is also shown: "`if (not(@mimeType) and not(parent::dash:AdaptationSet/@mimeType)) then false() else true()`". Line 38 shows the error message "`Either the Representation or the containing AdaptationSet shall have the mimeType attribute.`"

A.4.4.19 Example 18

`ex18.mpd` shows an invalid DASH MPD with the result in `result_ex18.xml`. The error occurs in the `SubRepresentation`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex18.mpd
     -o:output/result_ex18.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 28 to 31. Lines 28 and 29 show a failed assertion. The test which failed is also shown: "`if (@level and not(@bandwidth)) then false() else true()`". Line 30 shows the error message "`If the level attribute is defined for a SubRepresentation also the bandwidth attribute shall be defined.`"

A.4.4.20 Example 19

`ex19.mpd` shows an invalid DASH MPD with the result in result_ex19.xml. The error occurs in the `SegmentTemplate`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex19.mpd
     -o:output/result_ex19.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 29 to 32. Lines 29 and 30 show a failed assertion. The test which failed is also shown: "`if (not(@duration) and not(child::dash:SegmentTimeline)) then false() else true()`". Line 31 shows the error message "`If more than one Media Segment is present the duration attribute or SegmentTimeline element shall be present.`"

A.4.4.21 Example 20

`ex20.mpd` shows an invalid DASH MPD with the result in `result_ex20.xml`. The error occurs in the `SegmentTemplate`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex20.mpd
     -o:output/result_ex20.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 29 to 32. Lines 29 and 30 show a failed assertion. The test which failed is also shown: "`if (@duration and child::dash:SegmentTimeline) then false() else` true()". Line 31 shows the error message "Either the duration attribute or SegmentTimeline element shall be present but not both."

A.4.4.22 Example 21

ex21.mpd shows an invalid DASH MPD with the result in result_ex21.xml. The error occurs in the SegmentTemplate. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex21.mpd
     -o:output/result_ex21.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 29 to 32. Lines 29 and 30 show a failed assertion. The test which failed is also shown: "if (not(@indexRange) and @indexRangeExact) then false() else true()". Line 31 shows the error message "If indexRange is not present indexRangeExact shall not be present."

A.4.4.23 Example 22

ex22.mpd shows an invalid DASH MPD with the result in result_ex22.xml. There are two errors occurring in SegmentTemplate. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex22.mpd
     -o:output/result_ex22.xml -xsl:output/val_schema.xsl
```

The validation errors are shown in the lines 30 to 33 and lines 35 to 38. Lines 30 and 31 defined the first failed assertion. The test which failed is also shown: "if (@initialization and (matches(@initialization, '\$Number(%.[^\$]*)?\$') or matches(@initialization, ' \$Time(%.[^\$]*)?\$'))) then false() else true()". Line 32 shows the error message "Neither $Number$ nor the $Time$ identifier shall be included in the initialization attribute." Lines 35 and 36 defined the second failed assertion. The test which failed is also shown: "if (@initialization and (matches(@initialization, '\$Number(%.[^\$]*)?\$') or matches(@initialization, '\$Time(%.[^\$]*)?\$'))) then false() else true()". Line 37 shows the error message "Neither $Number$ nor the $Time$ identifier shall be included in the initialization attribute."

A.4.4.24 Example 23

ex23.mpd shows an invalid DASH MPD with the result in result_ex23.xml. There are two errors occurring in SegmentTemplate. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex23.mpd
     -o:output/result_ex23.xml -xsl:output/val_schema.xsl
```

The validation errors are shown in the lines 30 to 33 and lines 35 to 38. Lines 30 and 31 defined the first failed assertion. The test which failed is also shown: "if (@bitstreamSwitching and (matches(@bitstreamSwitching, '\$Number(%.[^\$]*)?\$') or matches(@bitstreamSwitching, '\$Time(%.[^\$]*)?\$'))) then false() else true()". Line 32 shows the error message "Neither $Number$ nor the $Time$ identifier shall be included in the bitstreamSwitching attribute." Lines 35 and 36 defined the second failed assertion. The test which failed is also shown: "if (@bitstreamSwitching and (matches(@bitstreamSwitching, '\$Number(%.[^\$]*)?\$') or matches(@bitstreamSwitching, '\$Time(%.[^\$]*)?\$'))) then false() else true()". Line 37 shows the error message "Neither $Number$ nor the $Time$ identifier shall be included in the bitstreamSwitching attribute."

A.4.4.25 Example 24

ex24.mpd shows an invalid DASH MPD with the result in result_ex24.xml. The error occurs in the SegmentList. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex24.mpd
     -o:output/result_ex24.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 30 to 33. Lines 30 and 31 show a failed assertion. The test which failed is also shown: "if (not(@duration) and not(child::dash:SegmentTimeline)) then if (count(child::dash:SegmentURL) &gt; 1) then false() else true() else true()". Line 32 shows the error message "If more than one Media Segment is present the duration attribute or SegmentTimeline element shall be present."

A.4.4.26 Example 25

ex25.mpd shows an invalid DASH MPD with the result in result_ex25.xml. The error occurs in the SegmentList. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex25.mpd
     -o:output/result_ex25.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 30 to 33. Lines 30 and 31 show a failed assertion. The test which failed is also shown: "if (@duration and child::dash:SegmentTimeline) then false() else true()". Line 32 shows the error message "Either the duration attribute or SegmentTimeline element shall be present but not both."

A.4.4.27 Example 26

ex26.mpd shows an invalid DASH MPD with the result in result_ex26.xml. The error occurs in the SegmentList. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex26.mpd
     -o:output/result_ex26.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 30 to 33. Lines 30 and 31 show a failed assertion. The test which failed is also shown: "if (not(@indexRange) and @indexRangeExact) then false() else true()". Line 32 shows the error message "If indexRange is not present indexRangeExact shall not be present."

A.4.4.28 Example 27

ex27.mpd shows an invalid DASH MPD with the result in result_ex27.xml. The errors occur in the SegmentBase and another in Representation. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex27.mpd
     -o:output/result_ex27.xml -xsl:output/val_schema.xsl
```

The validation errors are shown in the lines 26 to 29 and lines 36 to 39. Lines 26 and 27 defined the first
failed assertion. The test which failed is also shown: "if ((child::dash:SegmentBase and child::dash:SegmentTemplate and child::dash:SegmentList) or (child::dash:SegmentBase and child::dash:SegmentTemplate) or (child::dash:SegmentBase and child::dash:SegmentList) or (child::dash:SegmentTemplate and child::dash:SegmentList)) then false() else true()". Line 28 shows the error message "At most one of SegmentBase, SegmentTemplate and SegmentList shall be defined in Representation." Lines 36 and 37 defined the second failed assertion. The test which failed is also shown: "if (not(@indexRange) and @ indexRangeExact) then false() else true()". Line 38 shows the error message "If indexRange is not present indexRangeExact shall not be present."

A.4.4.29 Example 28

ex28.mpd shows an invalid DASH MPD with the result in result_ex28.xml. The error occurs in the SegmentTimeline. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex28.mpd
     -o:output/result_ex28.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 33 to 36. Lines 33 and 34 show a failed assertion. The test which failed is also shown: "if ((if (ancestor::dash:*[1]/@timescale) then `(child::dash:S/@d div ancestor::dash:*[1]/@timescale) else child::dash:S/@d) &gt; (years-from-duration(ancestor::dash:MPD/@maxSegmentDuration) + months-from-duration(ancestor::dash:MPD/@maxSegmentDuration) + days-from-duration(ancestor::dash:MPD/@maxSegmentDuration) + hours-from-duration(ancestor::dash:MPD/@maxSegmentDuration) + minutes-from-duration(ancestor::dash:MPD/@maxSegmentDuration) + seconds-from-duration(ancestor::dash:MPD/@maxSegmentDuration))) then false() else true()`". Line 35 shows the error message "`The d attribute of a SegmentTimeline shall not exceed the value give by the MPD maxSegmentDuration attribute.`"

A.4.4.30 Example 29

`ex29.mpd` shows an invalid DASH MPD with the result in `result_ex29.xml`. The error occurs in the `ProgramInformation`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex29.mpd
     -o:output/result_ex29.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 35 to 38. Lines 35 and 36 show a failed assertion. The test which failed is also shown: "`if (count(parent::dash:MPD/dash:ProgramInformation) &gt; 1 and not(@lang)) then false() else true()`". Line 37 shows the error message "`If more than one ProgramInformation element is given each ProgramInformation element shall have a lang attribute.`"

A.4.4.31 Example 30

`ex30.mpd` shows a valid DASH MPD with the result in `result_ex30.xml`. The error occurs in the `FramePacking`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex30.mpd
     -o:output/result_ex30.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 45 to 48. Lines 45 and 46 show a failed assertion. The test which failed is also shown: "`if (not(@value = '0' or @value = '1' or @value = '2' or @value = '3' or @value = '4' or @value = '5' or @value = '6')) then false() else true()`". Line 47 shows the error message "`The value of FramePacking shall be 0 to 6 as defined in` ISO/IEC 23091 (all parts)."

A.4.4.32 Example 31

`ex31.mpd` shows an invalid DASH MPD with the result in `result_ex31.xml`. The error occurs in the `ContentProtection`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex31.mpd
     -o:output/result_ex31.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 35 to 38. Lines 35 and 36 show a failed assertion. The test which failed is also shown: "`if ((@schemeIdUri = 'urn:mpeg:dash:mp4protection:2011') and not(string-length(@value) = 4)) then false() else true()`". Line 37 shows the error message "`The value of ContentProtection shall be the 4CC contained in the Scheme Type Box.`"

A.4.4.33 Example 32

`ex32.mpd` shows an invalid DASH MPD with the result in `result_ex32.xml`. The error occurs in the `ContentProtection`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex32.mpd
     -o:output/result_ex32.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 35 to 38. Lines 35 and 36 show a failed assertion. The test which failed is also shown: "`if ((@schemeIdUri = 'urn:mpeg:dash:13818:1:CA_descriptor:2011') and not(string-length(@value) = 4)) then false() else true()`". Line 37 shows the error message "`The value of ContentProtection shall be the 4-digit lower-case hexadecimal Representation.`"

A.4.4.34 Example 33

`ex33.mpd` shows an invalid DASH MPD with the result in `result_ex33.xml`. The error occurs in the `Role`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex33.mpd
     -o:output/result_ex33.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 36 to 39. Lines 36 and 37 show a failed assertion. The test which failed is also shown: "`if ((@schemeIdUri = 'urn:mpeg:dash:role:2011') and not(@value = 'caption' or @value = 'subtitle' or @value = 'main' or @value = 'alternate' or @value = 'supplementary' or @value = 'commentary' or @value = 'dub')) then false else true()`". Line 38 shows the error message "`The value of Role (role) shall be caption, subtitle, main, alternate, supplementary, commentary or dub.`"

A.4.4.35 Example 34

`ex34.mpd` shows an invalid DASH MPD with the result in `result_ex34.xml`. The error occurs in the `Role`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex34.mpd
     -o:output/result_ex34.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 37 to 40. Lines 37 and 38 show a failed assertion. The test which failed is also shown: "`if ((@schemeIdUri = 'urn:mpeg:dash:stereoid:2011') and not(starts-with(@value, 'l') or starts-with(@value, 'r'))) then false() else true()`". Line 39 shows the error message "`The value of Role (stereoid) shall start with 'l' or 'r'.`"

A.4.4.36 Example 35

`ex35.mpd` shows a valid DASH MPD with the result in `result_ex35.xml`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex35.mpd
     -o:output/result_ex35.xml -xsl:output/val_schema.xsl
```

A.4.4.37 Example 36

`ex36.mpd` shows an invalid DASH MPD with the result in `result_ex36.xml`. The error occurs in the `MPD`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex36.mpd
     -o:output/result_ex36.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 19 to 22. Lines 19 and 20 show a failed assertion. The test which failed is also shown: "`if (not(@profiles) or (contains(@profiles, 'urn:mpeg:dash:profile:isoff-on-demand:2011') or contains(@profiles, 'urn:mpeg:dash:profile:isoff-live:2011') or contains(@profiles, 'urn:mpeg:dash:profile:isoff-main:2011') or contains(@profiles, 'urn:mpeg:dash:profile:full:2011') or contains(@profiles, 'urn:mpeg:dash:profile:mp2t-main:2011') or contains(@profiles, 'urn:mpeg:dash:profile:mp2t-simple:2011'))) then true() else false()`".
Line 21 shows the error message "`The On-Demand profile shall be identified by the URN "urn:mpeg:dash:profile:isoff-on-demand:2011". The live profile shall be identified by the URN "urn:mpeg:dash:profile:isoff-live:2011". The main profile shall be identified by the URN "urn:mpeg:dash:profile:isoff-main:2011". The full profile shall be identified by the URN "urn:mpeg:dash:profile:full:2011". The mp2t-main profile shall be identified by the`

URN "urn:mpeg:dash:profile:mp2t-main:2011". The mp2t-simple profile shall be identified by the URN "urn:mpeg:dash:profile:mp2t-simple:2011"."

A.4.4.38 Example 37

ex37.mpd shows an invalid DASH MPD with the result in result_ex37.xml. There are two errors occurring, one in the MPD and one in the ContentProtection. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex37.mpd
     -o:output/result_ex37.xml -xsl:output/val_schema.xsl
```

The validation errors are shown in the lines 19 to 22 and lines 41 to 44. Lines 19 and 20 defined the first failed assertion. The test which failed is also shown: "if (not(contains(@profiles, 'urn:mpeg:dash:profile:isoff-on-demand:2011')) or not(@type) or @type='static') then true() else false()". Line 21 shows the error message "For On-Demand profile, the MPD @type shall be "static"." Lines 41 and 42 defined the second failed assertion. The test which failed is also shown: "if ((@schemeIdUri = 'urn:mpeg:dash:mp4protection:2011') and not(string-length(@value) = 4)) then false() else true()". Line 43 shows the error message "The value of ContentProtection shall be the 4CC contained in the Scheme Type Box."

A.4.4.39 Example 38

ex38.mpd shows an invalid DASH MPD with the result in result_ex38.xml. The error occurs in the ContentProtection. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex38.mpd
     -o:output/result_ex38.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 37 to 40 and lines 37 to 38. Lines 37 and 38 defined the failed assertion. The test which failed is also shown: "if ((@schemeIdUri = 'urn:mpeg:dash:mp4protection:2011') and not(string-length(@value) = 4)) then false() else true()". Line 39 shows the error message "The value of ContentProtection shall be the 4CC contained in the Scheme Type Box."

A.4.4.40 Example 39

ex39.mpd shows an invalid DASH MPD with the result in result_ex39.xml. The error occurs in the Representation. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex39.mpd
     -o:output/result_ex39.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 26 to 27. Lines 26 and 27 defined the failed assertion. The test which failed is also shown: "if (not(child::dash:SegmentTemplate or parent::dash:AdaptationSet/dash:SegmentTemplate or ancestor::dash:Period/dash:SegmentTemplate) and (contains(@profiles, 'urn:mpeg:dash:profile:isoff-live:2011') or contains(ancestor::dash:MPD/@profiles, 'urn:mpeg:dash:profile:isoff-live:2011') or contains(parent::dash:AdaptationSet/@profiles, 'urn:mpeg:dash:profile:isoff-live:2011'))) then false() else true()". Line 30 shows the error message "For live profile, 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."

A.4.4.41 Example 40

ex40.mpd shows an invalid DASH MPD with the result in result_ex40.xml. The error occurs in the SegmentTemplate. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex40.mpd
     -o:output/result_ex40.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 29 to 32. Lines 29 and 30 show a failed assertion. The test which failed is also shown: "if (matches(@media, '\$RepresentationID%.[^\$]*\$')) then false() else true()". Line 31 shows the error message "$RepresentationID$ shall not have a format tag."

A.4.4.42 Example 41

ex41.mpd shows an invalid DASH MPD with the result in result_ex41.xml. The error occurs in the Period. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex41.mpd
     -o:output/result_ex41.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 21 to 24. Lines 21 and 22 show a failed assertion. The test which failed is also shown: "if (not(descendant-or-self::dash:BaseURL) and not(descendant-or-self::dash:SegmentTemplate) and not(descendant-or-self::dash:SegmentList)) then false() else true()". Line 23 shows the error message "At least one BaseURL, SegmentTemplate or SegmentList shall be defined in Period, AdaptationSet or Representation."

A.4.4.43 Example 42

ex42.mpd shows an invalid DASH MPD with the result in result_ex42.xml. The error occurs in the AdaptationSet. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex42.mpd
     -o:output/result_ex42.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 23 to 26. Lines 23 and 24 show a failed assertion. The test which failed is also shown: "if ((child::dash:SegmentBase and child::dash:SegmentTemplate and child::dash:SegmentList) or (child::dash:SegmentBase and child::dash:SegmentTemplate) or (child::dash:SegmentBase and child::dash:SegmentList) or (child::dash:SegmentTemplate and child::dash:SegmentList)) then false() else true()". Line 25 shows the error message "At most one of SegmentBase, SegmentTemplate and SegmentList shall be defined in AdaptationSet."

A.4.4.44 Example 43

ex43.mpd shows a valid DASH MPD with the result in result_ex43.xml. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex43.mpd
-o:output/result_ex43.xml -xsl:output/val_schema.xsl
```

A.4.4.45 Example 44

ex44.mpd shows an invalid DASH MPD with the result in result_ex44.xml. The error occurs in the SegmentTemplate. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex44.mpd
     -o:output/result_ex44.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 29 to 32. Lines 29 and 30 show a failed assertion. The test which failed is also shown: "if (matches(@media, '\$.[^\$]*\$')) then every $y in (for $x in tokenize(@media, '\$(Bandwidth|Time|Number|RepresentationID)(%.[^\$]*)?\$') return matches($x, '\$.[^\$]*\$')) satisfies $y eq false() else true()". Line 31 shows the error message "Only identifiers such as $Bandwidth$, $Time$, $RepresentationID$, or $Number$ shall be used."

A.4.4.46 Example 45

ex45.mpd shows an invalid DASH MPD with the result in result_ex45.xml. The error occurs in the FramePacking. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex45.mpd
     -o:output/result_ex45.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 38 to 41. Lines 38 and 39 show a failed assertion. The test which failed is also shown: "if ((@schemeIdUri = 'urn:mpeg:dash:14496:10:frame_packing_arrangement_type:2011') and not(contains(parent::dash:AdaptationSet/@codecs, 'avc') or contains(parent::dash:AdaptationSet/@codecs, 'svc') or contains(parent::dash:AdaptationSet/@codecs, 'mvc')) and not(contains(parent::dash:Representation/@codecs, 'avc') or contains(parent::dash:Representation/@codecs, 'svc') or contains(parent::dash:Representation/@codecs, 'mvc'))) then false() else true()". Line 40 shows the error message "The URI urn:mpeg:dash:14496:10:frame_packing_arrangement_type:2011 is used for Adaptation Sets or Representations that contain a video component that conforms to ISO/IEC 14496-10."

A.4.4.47 Example 46

ex46.mpd shows an invalid DASH MPD with the result in result_ex46.xml. The error occurs in the FramePacking. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex46.mpd
     -o:output/result_ex46.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 38 to 41. Lines 38 and 39 show a failed assertion. The test which failed is also shown: "if ((@schemeIdUri = 'urn:mpeg:dash:13818:1:stereo_video_format_type:2011') and not(parent::dash:AdaptationSet/@mimeType = 'video/mp2t') and not(parent::dash:Representation/@mimeType = 'video/mp2t')) then false() else true()". Line 40 shows the error message "The URI urn:mpeg:dash:13818:1:stereo_video_format_type:2011 is used for Adaptation Sets or Representations that contain a video component that conforms to ISO/IEC 13818-1."

A.4.4.48 Example 47

ex47.mpd shows an invalid DASH MPD with the result in result_ex47.xml. The error occurs in the FramePacking. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex47.mpd
     -o:output/result_ex47.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 38 to 41. Lines 38 and 39 show a failed assertion. The test which failed is also shown: "if (not(@schemeIdUri = 'urn:mpeg:dash:14496:10:frame_packing_arrangement_type:2011') and not(@schemeIdUri = 'urn:mpeg:dash:13818:1:stereo_video_format_type:2011')) then false() else true()". Line 40 shows the error message "schemeIdUri for FramePacking descriptor shall be urn:mpeg:dash:14496:10:frame_packing_arrangement_type:2011 or urn:mpeg:dash:13818:1:stereo_video_format_type:2011."

A.4.4.49 Example 48

ex48.mpd shows an invalid DASH MPD with the result in result_ex48.xml. The error occurs in the FramePacking. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex48.mpd
     -o:output/result_ex48.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 38 to 41. Lines 38 and 39 show a failed assertion. The test which failed is also shown: "if (not(@value = '0' or @value = '1' or @value = '2' or @value = '3' or @value = '4' or @value = '5' or @value = '6')) then false() else true()". Line 40 shows the error message "The value of FramePacking shall be 0 to 6 as defined in ISO/IEC 23091 (all parts)".

A.4.4.50 Example 49

ex49.mpd shows an invalid DASH MPD with the result in result_ex49.xml. The error occurs in the AudioChannelConfiguration. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex49.mpd
     -o:output/result_ex49.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 39 to 42. Lines 39 and 40 show a failed assertion. The test which failed is also shown: "`if ((@schemeIdUri = 'urn:mpeg:dash:outputChannelPositionList:2012') and not(count(tokenize(@value, ' ')) &gt; 1)) then false() else true()`". Line 41 shows the error message "`If URI urn:mpeg:dash:outputChannelPositionList:2012 is used the value attribute shall be a space-delimited list as defined in` ISO/IEC 23091 (all parts)".

A.4.4.51 Example 50

ex50.mpd shows an invalid DASH MPD with the result in result_ex50.xml. The error occurs in the SegmentTemplate. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex50.mpd
     -o:output/result_ex50.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 19 to 22. Lines 19 and 20 show a failed assertion. The test which failed is also shown: "`if (@type = 'dynamic' and not(@publishTime)) then false() else true()`". Line 21 shows the error message "`If MPD is of type "dynamic" publishTime shall be defined.`"

A.4.4.52 Example 51

`ex51.mpd` shows an invalid DASH MPD with the result in `result_ex51.xml`. The error occurs in the `SegmentTemplate`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex51.mpd
-o:output/result_ex51.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 41 to 44. Lines 41 and 42 show a failed assertion. The test which failed is also shown: "`if (@id = preceding-sibling::dash:Subset/@id) then false() else true()`". Line 43 shows the error message "`The id of each Subset within a Period shall be unique.`"

A.4.4.53 Example 52

`ex52.mpd` shows an invalid DASH MPD with the result in `result_ex52.xml`. The error occurs in the `EventStream`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex52.mpd
     -o:output/result_ex52.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 41 to 44. Lines 41 and 42 show a failed assertion. The test which failed is also shown: "`if ((@schemeIdUri = preceding::dash:EventStream/@schemeIdUri) and (@value = preceding::dash:EventStream/@value)) then false() else true()`". Line 43 shows the error message "`A Period shall contain at most one EventStream element with the same value for the schemeIdUri attribute and value attribute.`"

A.4.4.54 Example 53

`ex53.mpd` shows an invalid DASH MPD with the result in `result_ex53.xml`. The error occurs in the `EventStream`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex53.mpd
     -o:output/result_ex53.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 40 to 43. Lines 40 and 41 show a failed assertion. The test which failed is also shown: "`if ((@schemeIdUri = 'urn:mpeg:dash:event:2012') and not(@value = '1' or @value = '2')) then false() else true()`". Line 42 shows the error message "`If URI urn:mpeg:dash:event:2012 is used the value attribute shall have the value 1 or 2.`"

A.4.4.55 Example 54

`ex54.mpd` shows an invalid DASH MPD with the result in `result_ex54.xml`. The error occurs in the `InbandEventStream`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex54.mpd
     -o:output/result_ex54.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 42 to 45. Lines 42 and 43 show a failed assertion. The test which failed is also shown: "`if ((@schemeIdUri = preceding::dash:InbandEventStream/@schemeIdUri) and (@value = preceding::dash:InbandEventStream/@value)) then false() else true()`". Line 44 shows the error message "`There shall be only one InbandEventStream element with the same value for the schemeIdUri attribute and value attribute.`"

A.4.4.56 Example 55

`ex55.mpd` shows an invalid DASH MPD with the result in `result_ex55.xml`. The error occurs in the `InbandEventStream`. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex55.mpd
     -o:output/result_ex55.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 41 to 44. Lines 41 and 42 show a failed assertion. The test which failed is also shown: "`if ((@schemeIdUri = 'urn:mpeg:dash:event:2012') and not(@value = '1' or @value = '2')) then false() else true()`". Line 43 shows the error message "`If URI urn:mpeg:dash:event:2012 is used the value attribute shall have the value 1 or 2.`"

A.4.4.57 Example 56

`ex56.mpd` shows an invalid DASH MPD with the result in `result_ex56.xml`. The error occurs in the Event. The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/ex56.mpd
     -o:output/result_ex56.xml -xsl:output/val_schema.xsl
```

The validation error is shown in the lines 46 to 49. Lines 46 and 47 show a failed assertion. The test which failed is also shown: "`if (if (not(@presentationTime)) then (0 &lt; preceding-sibling::dash:Event/@presentationTime) else (@presentationTime &lt; preceding-sibling::dash:Event/@presentationTime)) then false() else true()`". Line 48 shows the error message "`Events shall be in non-decreasing order based on the presentation time.`"

A.5 Step 4: Segment validator

A.5.1 ISO base media file format segments

A.5.1.1 General

The ISO base media file format validator software has two parts: a software backend and a web-based frontend. The only input required to generate the conformance results is an MPD that can either be hosted on a web-server or stored locally on the file system.

A.5.1.2 Generating conformance results

In order to generate the conformance results, the user-interface of the web-based front-end is used to provide the URL or local file system location of the MPD to be verified. The processing runs automatically in the following steps.

a) The MPD is loaded.

b) The MPD is parsed and processed to extract relevant attributes and determine the segment URLs.

c) A small amount of data is downloaded from each segment to re-create the relevant parts of the representation at the server running the conformance software. ‘ftyp’, ‘moov’, ‘styp’, ‘sidx’, ‘ssix’, and ‘moof’ are the box types included in the downloaded data. The headers of any ‘mdat’ boxes are also downloaded.

d) Each representation is tested following the conformance checks documented in the previous sections, including cross-representation conformance checks.

e) The user interface indicates if the segments are conforming or not. If they are not, an error report is linked in the user interface. Opening the error report provides the description of each conformance error found in relation with the specification.

f) The cached data used for processing is deleted automatically without user intervention.

In general, operating the user interface is self-explanatory, with descriptions provided to allow for completion of conformance testing. Each conformance test run is isolated in its own instance, allowing for parallel concurrent testing.

A.5.2 MPEG-2 transport stream segments

A.5.2.1 General

The ts_validator verifies:

— MPEG transport stream syntax and conformance;

— SAP type conformance;

— Segment timing (for audio and video);

— Index file conformance;

— Initialization segment conformance;

— Bitstream switching segment conformance.

The ts_validator reads all required metadata directly from the MPD.

To run the ts_validator, open a shell for a command line, and run ‘ts_validate_mult_segment’ with the MPD as an input parameter. For example:

```
ts_validate_mult_segment example.mpd
```

The validator will produce a long report. To save the report to file, use file redirection:

```
ts_validate_mult_segment example.mpd > myTestOutput.txt
```

The report contains a lot of information, including pass/fail information for each initialization segment, index segment, and media segment. Below we provide more detail on a few of the features.

By default, only conformance problems, warnings and errors are written to the output. If you want more information, run with -v for info-level logging and -vv for debug level.

A.5.2.2 Individual segment results

The report contains a pass/fail result for each media segment. To find these, search for the text "SEGMENT TEST RESULT". If a segment test has failed, then more info on the failure will be provided in the lines preceding the "SEGMENT TEST RESULT" line.

A.5.2.3 Overall test result

The last line of the report contains a summary pass/fail test result. To find this, either scroll to the end of the report or search for the text "OVERALL TEST RESULT".

A.5.2.4 Audio and video gap matrices

The report contains audio and video timing gap matrices in order to measure the timing gaps that a user would experience if switching between representations. Examples are shown below. For example, if a user switches from uf7/seg-00001.ts to uf12/seg-00002.ts, then the timing disruption they would experience can be found in row uf7/seg-00001.ts, column uf12/seg-00002.ts of the timing matrix. In the example below, all of the timing gaps are 0, so the user will not experience any timing disruption when switching between these segments. The test report contains matrices for all segment pairs in an adaptation set. The matrices are tab-delimited, so they look clumsy in the test report, but display nicely if pasted into a spreadsheet.

AudioGapMatrix

	uf3/seg-00002.ts	uf7/seg-00002.ts	uf12/seg-00002.ts	uf28/seg-00002.ts
uf3/seg-00001.ts	0	0	0	0
uf7/seg-00001.ts	0	0	0	0
uf12/seg-00001.ts	0	0	0	0
uf28/seg-00001.ts	0	0	0	0

VideoGapMatrix

	uf3/seg-00002.ts	uf7/seg-00002.ts	uf12/seg-00002.ts	uf28/seg-00002.ts
uf3/seg-00001.ts	0	0	0	0
uf7/seg-00001.ts	0	0	0	0
uf12/seg-00001.ts	0	0	0	0
uf28/seg-00001.ts	0	0	0	0

A.5.2.5 Audio and video timing summaries

A tabular summary of segment timings for each representation is given near the end of the test report, and an example is shown below. The table shows the expected and actual start and end times for each segment (in PTS ticks). The matrices are tab-delimited, so they look clumsy in the test report, but display nicely if pasted into a spreadsheet.

VideoTiming

segmentFile	expectedStart	expectedEnd	videoStart	videoEnd	deltaStart	deltaEnd
uf28/seg-00001.ts	6000	906000	6000	906000	0	0
uf28/seg-00002.ts	906000	1806000	906000	1806000	0	0
uf28/seg-00003.ts	1806000	2706000	1806000	2706000	0	0
uf28/seg-00004.ts	2706000	3606000	2706000	3606000	0	0

uf28/seg-00005.ts	3606000	4506000	3606000	4506000	0	0
uf28/seg-00006.ts	4506000	5406000	4506000	5406000	0	0
uf28/seg-00007.ts	5406000	6306000	5406000	6306000	0	0
uf28/seg-00008.ts	6306000	7206000	6306000	7206000	0	0
uf28/seg-00009.ts	7206000	8106000	7206000	8106000	0	0
uf28/seg-00010.ts	8106000	9006000	8106000	9006000	0	0
uf28/seg-00011.ts	9006000	9906000	9006000	9906000	0	0
uf28/seg-00012.ts	9906000	10806000	9906000	10806000	0	0
uf28/seg-00013.ts	10806000	11706000	10806000	11706000	0	0
uf28/seg-00014.ts	11706000	12606000	11706000	12606000	0	0
uf28/seg-00015.ts	12606000	13506000	12606000	13506000	0	0
uf28/seg-00016.ts	13506000	14406000	13506000	14406000	0	0
uf28/seg-00017.ts	14406000	15306000	14406000	15306000	0	0
uf28/seg-00018.ts	15306000	16206000	15306000	16206000	0	0
uf28/seg-00019.ts	16206000	17106000	16206000	17106000	0	0
uf28/seg-00020.ts	17106000	18006000	17106000	18006000	0	0
uf28/seg-00021.ts	18006000	18906000	18006000	18906000	0	0
uf28/seg-00022.ts	18906000	19806000	18906000	19806000	0	0
uf28/seg-00023.ts	19806000	20706000	19806000	20706000	0	0
uf28/seg-00024.ts	20706000	21606000	20706000	21606000	0	0
uf28/seg-00025.ts	21606000	22506000	21606000	22506000	0	0
uf28/seg-00026.ts	22506000	23153520	22506000	23151000	0	-2520

The following test vectors conform to the ISO BMFF on-demand profile.

Features	Comment
5) Test Vector(1): association-1.mpd	
6) Contains one metadata Representation associated with a video Representation using associationId alone (no associationType)	
7) Test Vector(2): association-2.mpd	
8) Contains one metadata Representation associated with multiple video Representations using multiple associationId values.	
9) Test Vector(3): association-3.mpd	
10) Contains one metadata Representation associated with a video Representation using associationId and associationType, each having one value	

Features	Comment
11) Test Vector(4): association-4.mpd	
12) Contains one metadata Representation associated with multiple video Representations using associationId and associationType, each having multiple values (same number)	
13) Test Vector (5): association_signalling1.mpd	
14) In this example, a metadata Representation in an Adaptation Set is associated with a media Representation in another Adaptation Set for video. Each metadata Segment corresponds to a media Segment.	
15) Test Vector (6) Association-greenMetadata.mpd contains a specific Adaptation Set within the MPD that defines the available Green Metadata representations and their association with the available video representations. More description available in m34934.	

The following test vector conforms to the ISO base media file format live profile.

Features	Comment
16) Test Vector (1): association_signalling2.mpd	
17) In this example, a metadata Representation in an Adaptation Set is associated with media Representations in another Adaptation Set for video. The metadata Representation multiplexes timed metadata tracks carrying quality information for the associated media Representations	

The additional test vectors below provide invalid test cases.

Features	Comment
18) Test Vector(8) Association-invalid1.mpd: one metadata Representation is associated with a Representation that does not exist. This is not allowed	
19) Test Vector (9) Association-invalid2.mpd: one metadata Representation has an associationType without any associationId. This is not allowed	
20) Test Vector(10) Association-invalid3.mpd: one metadata Representation is associated with two video Representations but has only one value in its associationType. This is not a valid MPD (associationType should have same number as associationId).	
21) Test Vector(11) Association-invalid4.mpd: one metadata Representation is associated with two Representations. One of the two referenced Representation is not present in the MPD. This is not a valid MPD.	
22) Test Vector(12) Association-invalis5.mpd: one metadata Representation, associated with a video Representation, has an associationType value that is not a four character string. This is not a valid MPD.	

A.5.2.5.1 Example association-1

23) association-1.mpd shows a valid DASH MPD with the result in result_association-1.xml

The command used is as follows:

```
java -jar saxon/saxon9.jar -versionmsg:off -s:examples/association-1.mpd
     -o:output/result_association-1.xml -xsl:output/val_schema.xsl
```

A.5.2.5.2 Example association-2

24) association-2.mpd shows a valid DASH MPD with the result in result_association-2.xml

The command used is as follows: