** ISO/IEC JTC 1/SC 29/WG 03 N0734**

**ISO/IEC JTC 1/SC 29/WG 03  
MPEG Systems   
Convenorship: KATS (Korea, Republic of)**

**Document type:** Output Document

**Title:** Draft text of ISO/IEC 23009-1 5th edition FDAM 1 Alternative MPD event, nonlinear playback and other extensions

**Status:** Approved

**Date of document:** 2022-10-28

**Source:** ISO/IEC JTC 1/SC 29/WG 03

**Expected action:**  None

**Action due date:** None

**No. of pages:** 44 (with cover page)

**Email of Convenor:** young.L @ samsung . com

**Committee URL:** <https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg3>

**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC1/SC29/WG11**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC 1/SC 29/WG 03 N** **0734**

**January 2022, Virtual**

|  |  |
| --- | --- |
| **Title** | **Draft text of ISO/IEC 23009-1 5th 4th edition FDAM 1 Alternative MPD event, nonlinear playback and other extensions** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **22013** |

**ISO/IEC 23009-1:2022/DAM2**

ISO/IEC TC JTC 1/SC 29/WG 11

Secretariat: JISC

**Information technology —** **Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats, Amendment 1: Alternative MPD event, nonlinear playback and other extensions**

**Warning**

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

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

© ISO 2018, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO’s member body in the country of the requester.

ISO copyright office

Ch. de Blandonnet 8 • CP 401

CH-1214 Vernier, Geneva, Switzerland

Tel.  + 41 22 749 01 11

Fax  + 41 22 749 09 47

copyright@iso.org

www.iso.org

# 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](https://www.iso.org/directives-and-policies.html) or [www.iec.ch/members\_experts/refdocs](https://www.iec.ch/members_experts/refdocs)).

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](https://www.iso.org/iso-standards-and-patents.html)) or the IEC list of patent declarations received (see <https://patents.iec.ch>).

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](https://www.iso.org/iso/foreword.html). In the IEC, see [www.iec.ch/understanding-standards](https://www.iec.ch/understanding-standards).This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

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

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at [www.iso.org/members.html](https://www.iso.org/members.html) and [www.iec.ch/national-committees](https://www.iec.ch/national-committees).

**Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats, Amendment 1: Alternative MPD event, nonlinear playback and other extensions**

*3.1*

*Add a new entry at the end of the subclause:*

**3.1.54**  
**Preroll**  
on-demand content which is played before start of playback of a live content

**3.1.X**  
**Media Presentation Insertion**

Insertion of media content from a second media presentation in the timeline of a first media presentation

*4.7, Table 2*

Add the following rows to Table 2:

|  |  |  |
| --- | --- | --- |
| urn:mpeg:dash:event:period:2020 | 5.10.4.7 | Scheme Identifier for signalling the Period events |
| urn:mpeg:dash:event:alternative:2022 | 5.13 | Scheme Identifier for alternative MPD event |
| urn:mpeg:dash:nonlinearplayback:2020 | K.3.2 | Scheme Identifier for signalling nonlinear storyline events |

*5.10.4.2*

*Change the following item:*

— the event duration expresses the remaining duration of Media Presentation from the event time. If the event duration is 0, the Media Presentation ends at the event time. If 0xFFFF, the media presentation duration is unknown.

*to:*

— the event duration expresses the remaining duration of Media Presentation from the event time. If the event duration is 0, the Media Presentation ends at the event time. If 0xFFFF, the media presentation duration is unknown. This feature is deprecated.

*5.16*

*Add the following new subclause after subclause 5.15 and before Clause 6:*

## 5.16 Media Presentation Insertion

### 5.16.1 General

Media Presentation Insertion provides a mechanism to indicate the insertion of an alternative Media Presentation into the main Media Presentation at a certain point of the Media Presentation timeline. After the playback of the alternative Media Presentation, the playback is continued with the main Media Presentation. The alternative Media Presentation may be inserted in the media time of the Main Presentation, i.e. time-shift the part of the Main Presentation which is played after the alternative Media Presentation, or it may replace a time segment of the main Media Presentation.

After the playback of the alternative Media Presentation, the playback of the main Media Presentation is typically be continued. Transitioning between the two Media Presentations is expected to be without any latencies, i.e. the timelines of the main Media Presentation and the inserted Media Presentation are sequential with no gaps during playback.

This tool provides the ability to switch between two independent Media Presentations, for applications such as preroll and mid-roll during a live-streaming session. It may also be used to insert content such as ads into an On-Demand Media Presentation.

The alternative Media Presentation, the start time of the alternative Media Presentation, and the rejoin time to the main Media Presentation are signaled using a DASH event scheme. While the general processing of this event is similar to other DASH events, a specific post-processing procedure is defined in this clause for transition between the two timelines during the playback.

### 5.16.2 Alternative MPD event

This event scheme is identified by the URN "urn:mpeg:dash:event:alternativeMPD:2022". The event scheme’s dispatch mode is on-receive.

Table 52 defines the relevant MPD parameters for the alternative MPD event.

Table 52 — Relevant parameters for alternative MPD event in MPD

| **Attribute** | **Value** |
| --- | --- |
| EventStream@schemeIdUri/  InbandEventStream@schemeIdUri | "urn:mpeg:dash:event:alternative:2022" |
| EventStream@value/  InbandEventStream@value | * + “replace”: to return to (the time in Media Presentation in which the playback is being switched to the alternative Presentation + the alternative Media Presentation duration) or the end of Media Presentation whichever is earlier.   + “insert”: to return to the moment in Media Presentation that the playback is switched to the alternative Media Presentation |
| Event@presentationTime | The earliest presentation time that switching from the main Media Presentation to the alternative Media Presentation occurs. |
| Event@duration | The duration over which the switch from the main Media Presentation to the alternative Media Presentation occurs. . |
| Event body | URL of the MPD representing the alternative Media Presentation.  If this URL does not resolve to a valid MPD, the playback is continued with the main Media Presentation. |

Table 53 shows the relevant emsg parameters for alternative MPD events.

Table 53— Relevant emsg parameters for alternative MPD event

| **Attribute** | **Value** |
| --- | --- |
| scheme\_id\_uri | "urn:mpeg:dash:event:alternative:2022" |
| value | * + “replace”: to return to (the time in Media Presentation in which the playback is being switched to the alternative Presentation + the alternative Media Presentation duration) or the end of Media Presentation whichever is earlier.   + “insert”: to return to the moment in Media Presentation that the playback is switched to the alternative Media Presentation |
| presentation\_time\_delta/  prsentation\_time | The offset in which the event for Media Presentation Insertion becomes active. The anchor for each offset parameter is defined in 5.10.3.2 accordingly. |
| event\_duration | The duration in which the event for Media Presentation Insertion is active. |
| message\_data | The MPD URL of the alternative Media Presentation.  If this URL does not resolve to a valid MPD, the playback is continued with the main Media Presentation. |

*A.14*

### *Add the following subclause after A.13.12 and before Annex B:*

### A.14 Alternative MPD event post-processing model

The Alternative MPD event is processed and dispatched according to clause A.13. This clause defines the post-processing of this event after being dispatched. This clause is informative and is intended to show the expected behavior from the DASH client.

The post-processing procedure of the event relies on the parameters shown in Table A.4.

Table A.4 Event/timed metadata API parameters and datatypes

| **API Parameter** | **MPD event** | **Inband event** | **values** |
| --- | --- | --- | --- |
| scheme\_id | **EventStream**@schemeIdUri | scheme\_id\_uri | "urn:mpeg:dash:event:alternative:2022" |
| value | **EventStream**@value | value | * + “replace”: to return to (the time in Media Presentation in which the playback is being switched to the alternative Presentation + the alternative Media Presentation duration) or end of presentation whichever is earlier.   + “insert”: to return to the moment in Media Presentation that the playback is switched to the alternative Media Presentation |
| presentation\_time | **Event**@presentationTime | presentation\_time | The offset in which the event for Media Presentation Insertion becomes active |
| duration | **Event@**duration | duration | The duration in which the event for Media Presentation Insertion is active |
| message | **Event** body | message\_data | The MPD URL of the alternative Media Presentation. |

The client alternative MPD switching event post-processing procedure is as the following:

1. The client checks if the alternative MPD URL in message is in its Previously Played List (PPL). If so, it doesn’t take any further action, otherwise, it continues the following steps.
2. It downloads the alternative MPD.
3. One of the following cases:
   1. If presentation\_time ≤ current playback time ≤ presentation\_time, it immediately goes to step d.
   2. If current playback time < presentation\_time, it continues playback of the main Media presentation until the current playback time = presentation\_time.
4. Set switch\_time = current playback time. Then, it switches the playback from the main Media Presentation to the alternative Media Present as long as the main Media Presentation is not ended. Otherwise, it stops and clears its switch\_time and PLL buffers.
5. It stores the main MPD URL and switch\_time .
6. It adds the message to its PPL.
7. At the end of the alternative Media Presentation, it downloads the main MPD from the main MPD URL.
8. It continues playing back the main Media Presentation according to the value of value:
   1. If value =’replace’, from (switch\_time + duration of alternative Media Presentation) or the end of Media Presentation whichever is earlier.
   2. If value =’insert’, from switch\_time\*.

Note 1: The DASH client clears its URL, switch\_time, PPL values starting at the first parsing of the main MPD and continues maintaining them during the entire playback.

Note 2: The exact time of switching, switch\_time, depends on how the players reaches the active time interval, e.g. by linear playback to its start time, or by random access to a moment in the middle of it.

Note 3: In the case of \*, setting @timeshiftBufferDepth to a value equal to or larger than the maximum alternative Media Presentation duration assures that the media segments would be available at switch\_time when playback is returned to the main Media Presentation.

*G.23, G.24 and G.25*

*Add the following subclause after G.22 and before Annex H:*

**G.23 Insert Preroll at the beginning of a live content**

An example of inserting one Preroll ad at the beginning of each playback of a live content using the alternative MPD event is provided below.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <MPD     xmlns="urn:mpeg:dash:schema:mpd:2011"      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"      xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 DASH-MPD.xsd"    availabilityStartTime="1970-01-01T00:00:00Z" maxSegmentDuration="PT6S"    minBufferTime="PT2S" minimumUpdatePeriod="PT5M"    profiles="urn:mpeg:dash:profile:isoff-live:2011" publishTime="2019-03-12T01:17:30Z"    timeShiftBufferDepth="PT8M20S" type="dynamic">    <Period id="p0" start="PT0S">        <EventStream schemeIdUri="urn:mpeg:dash:event:alternative:2022S" value="replace">  < Event <http://acmeadsertver.com/preroll.mpd> presentationTime="PT0S"  duration=="10000000"/>  </EventStream>        <BaseURL>http://liveserver.com/live/live1/</BaseURL>        <AdaptationSet contentType="video" maxHeight="1920" maxWidth="1080"            mimeType="video/mp4" par="16:9" segmentAlignment="true" startWithSAP="1">           <SegmentTemplate duration="2" initialization="$RepresentationID$/init.mp4"              media="$RepresentationID$/$Number$.m4s" startNumber="0" />           <Representation id="V300" bandwidth="300000" codecs="avc1.64001e"              frameRate="60/2" />           <Representation id="V600" bandwidth="600000" codecs="avc1.64001e"              frameRate="60/2" />        </AdaptationSet>    </Period>  </MPD> |

In this example, when the client starts playing the live content, first it plays the Preroll content represented by the MPD obtained from <http://acmeadsertver.com/preroll.mpd> and then it plays the live streaming session represented by Period with @id=’p0’ by joining the live edge of this media presentation.

**G.24 Insert one midroll in the live content**

An example of inserting one midroll 1 minute after the start of Period using the alternative MPD event is provided below.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <MPD     xmlns="urn:mpeg:dash:schema:mpd:2011"      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"      xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 DASH-MPD.xsd"    availabilityStartTime="1970-01-01T00:00:00Z" maxSegmentDuration="PT6S"    minBufferTime="PT2S" minimumUpdatePeriod="PT5M"    profiles="urn:mpeg:dash:profile:isoff-live:2011" publishTime="2019-03-12T01:17:30Z"    timeShiftBufferDepth="PT8M20S" type="dynamic">    <Period id="p0" start="PT0S">        <EventStream schemeIdUri="urn:mpeg:dash:event:alternative:2022S" value="replace">  < Event <http://acmeadsertver.com/preroll.mpd> presentationTime="PT60S"  duration="10000000"/>  </EventStream>        <BaseURL>http://liveserver.com/live/live1/</BaseURL>        <AdaptationSet contentType="video" maxHeight="1920" maxWidth="1080"            mimeType="video/mp4" par="16:9" segmentAlignment="true" startWithSAP="1">           <SegmentTemplate duration="2" initialization="$RepresentationID$/init.mp4"              media="$RepresentationID$/$Number$.m4s" startNumber="0" />           <Representation id="V300" bandwidth="300000" codecs="avc1.64001e"              frameRate="60/2" />           <Representation id="V600" bandwidth="600000" codecs="avc1.64001e"              frameRate="60/2" />        </AdaptationSet>    </Period>  </MPD> |

In this example, after playback of 1 minute of the live content in Period with @id=’p0’, the playback is switched to playing the midroll content with the corresponding MPD obtained from <http://acmeadsertver.com/preroll.mpd>. Then, the playback is continued by rejoining the live streaming session at its live edge represented.

**G.25 Insert one midroll in the live content with a time shift**

An example of inserting one midroll 1 minute after the start of Period using the alternative MPD event is provided below.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <MPD     xmlns="urn:mpeg:dash:schema:mpd:2011"      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"      xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 DASH-MPD.xsd"    availabilityStartTime="1970-01-01T00:00:00Z" maxSegmentDuration="PT6S"    minBufferTime="PT2S" minimumUpdatePeriod="PT5M"    profiles="urn:mpeg:dash:profile:isoff-live:2011" publishTime="2019-03-12T01:17:30Z"    timeShiftBufferDepth="PT8M20S" type="dynamic">    <Period id="p0" start="PT0S">        <EventStream schemeIdUri="urn:mpeg:dash:event:alternative:2022S" value="insert">  < Event <http://acmeadsertver.com/preroll.mpd> presentationTime="PT60S"  duration=="0"/>  </EventStream>        <BaseURL>http://liveserver.com/live/live1/</BaseURL>        <AdaptationSet contentType="video" maxHeight="1920" maxWidth="1080"            mimeType="video/mp4" par="16:9" segmentAlignment="true" startWithSAP="1">           <SegmentTemplate duration="2" initialization="$RepresentationID$/init.mp4"              media="$RepresentationID$/$Number$.m4s" startNumber="0" />           <Representation id="V300" bandwidth="300000" codecs="avc1.64001e"              frameRate="60/2" />           <Representation id="V600" bandwidth="600000" codecs="avc1.64001e"              frameRate="60/2" />        </AdaptationSet>    </Period>  </MPD> |

In this example, after playback of 1 minute of the live content in Period with @id=’p0’, the playback is switched to the midroll content with the corresponding MPD obtained from t<http://acmeadsertver.com/preroll.mpd>. After playback of the midroll content, the playback is continued from the moment in the live streaming session that was switched to the midroll content.

*Annex L*

*Add the following annex after Annex K, before the Bibliography*

**Annex L**

(normative)

**Implementation of Nonlinear Playback**

* 1. General

This Annex provides Nonlinear Playback capabilities, to serve Interactive Storyline content with MPEG-DASH. It consists of the normative requirements for MPD authoring and a recommended behaviour for the DASH client that supports the Nonlinear Playback defined in this Annex.

* 1. Overview

Typically, a DASH Client consumes the Periods contained in an MPD in a single timeline in a serial (linear) fashion. The Interactive Storyline use-case concerns dynamic nonlinear playback of the content Periods. The application makes decisions upon which the user selects which Period to consume after the end of the currently active period. Conceptually, this behaviour can be represented as a directed acyclic graph as in Figure 1, where the graph edges (shown by double-lined arrows) represent the different Periods Periods and the graph nodes (S0, S1 …) contain the available decisions (shown by single line arrows) i.e., the end of the Periods and the possible next Periods to playback. The logic upon the decisions are made is application-specific thus out of the scope of this text.

Figure 1 Example of Interactive Storyline graph

Start

End

P0

P1

P2

P4

P5

P6

P7

S0

S1

P3

Figure 1 Example of Interactive Storyline graph

In this graph, the “available” Period is the Periods available for selection at any graph nodes, and the “selected” Period is the “available” Period that is selected by the Application to be played among all “available” Period at that node.

* 1. Description of Elements Required for Nonlinear Storyline

To be conformant to this Annex, this section describes how the Period, Event, and EventStream elements are used for nonlinear storyline implementation.

* + 1. Periods for Nonlinear Storyline

Each Period element represents an edge of the graph and shall adhere to the following guidelines:

* It should have the @duration attribute present and equal to the length of the graph edge it represents.
* It shall contain a storyline EventStream element as described in 5.10.2.1.
  + 1. Selection Parameters

The DASH Client signals the chosen edge to play after the end of the current by firing a callback at the @contactUrl of the respective Event and signalling the selection as the query parameters.

Examples of the callback are:

* Call back at end of Period@id=1: http://cdn.com/content\_xyz/1/selection? parameter=${param-value}.
* Call back at end of Period@id=2: http://cdn.com/content\_xyz/2/selection? parameter=${param-value}.

This way, the HTTP server receiving the callback request can determine the Period the DASH Client is currently playing back from the **Period**@id in the URL.

* + 1. Events for Nonlinear Storyline

The Event element is extended with a parametrization scheme to accommodate the selection information (nlp:selectionInfo) - feature specific to the nonlinear storyline. The scheme is identified by URN “urn:mpeg:dash:nonlinearplayback:2020”.

The extension for the Event element of the namespace is defined in A.3.3.1 and of the **SelectionInfo** in A.3.3.2.

* + - 1. The Event Element

Table 2 — Event Semantics

| **Element or Attribute Name** | | | | **Use** | **Description** |
| --- | --- | --- | --- | --- | --- |
|  |  | Event | |  |  |
|  |  |  | @nlp:selectionInfo | O | specifies the value for the event stream element. This attribute of the Event carries interactive data about the selection point. |
|  |  |  | @contactURL | 1 | specifies the URL to which an HTTP GET request is expected to be issued after adding ‘/${**Period**@id}/selection?parameter=${**SelectionInfo**@parameter}’, where ${@attribute} is the value of @attribute and **Period** and **SelectionInfo** are the current Period and the SelectionInfo element corresponding to the selected Period.  The URL shall be a NULL-terminated string.  HTTP response shall either not be provided or be provided such that it can be discarded. |
|  |  |  | SelectionInfo | 1 ... N | specifies parameter and data in the query |

When the namespace is set as “urn:mpeg:dash:nonlinearplayback:2020”, the attribute @nlp:selectionInfo is used to provide to the client information on the possible selections. For the same namespace, the Event attributes @presentationTime and @duration are used to indicate the start and the length of the selection window, respectively.

The @messageData or the body of the Event is used to carry the data common for all choices in this selection. For example, the title of choice can be carried in the Event body.

* + - 1. SelectionInfo Element

Table 3 — SelctionInfo Semantics

| **Element or Attribute Name** | | | | **Use** | **Description** |
| --- | --- | --- | --- | --- | --- |
|  |  | SelectionInfo | |  |  |
|  |  |  | @dataEncoding | O | specifies whether the information in the body and the information in the @data is encoded.  If present, the following value is possible:   * base64 the content is encoded as described in IETF RFC 4648 prior to adding it to the field.   If this attribute is present, the DASH Client is expected to decode the message data and only provide the decoded message to the application. |
|  |  |  | @parameter | M | identifier of the available Period. This value shall be unique among @parameter attributes in the scope of this **SelectionInfo** element. |
|  |  |  | @data | O | interactive data for the application for the @parameter period. The interactive data may exist in the body, and if it does, it takes precedent to this attribute. |
| **Key**  For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory  For elements: <minOccurs>...<maxOccurs> (N=unbounded)  Elements are bold; attributes are non-bold and preceded with an @. | | | | | |

Note that the @data value is opaque to the DASH client and only if it is Base64 encoded, is decoded by the DASH client and delivered to the application.

The format of the @data for the application (as well as **Event**@messageData) may be signalled by inserting a URL in the **Period**.**EventStream**@value, i.e a DASH client is expected to find information about the format of these attributes at this URL.

The available parameter values for the callback are expressed via the @parameter attributes of the **SelectionInfo** element. It may contain @id of the available Periods, or custom parameters for the available Periods that can be identified by the server, or a mix of both. As a result, an available Period can be expressed by one or more values.

For explicit mapping onto Periods, the value of the selection’s query parameter shall hold one of the @id of the possible next Periods to playback. This way, the HTTP server receiving the callback request can determine directly the next Period to add to the next MPD version directly from the selection query parameter value. Alternatively, the selection query parameter value may be indicative of the graph edge and a further translation needs to be made to identify the Period to insert in the next MPD version. Note that the DASH Client in both cases is not aware of the meaning of the value reported. This is only known to the DASH server.

To use the @contactUrl for the signaling selected Period, the selection parameter is defined as the respective SelectionInfo attribute (i.e., parameter) and the selection value is the value of the attribute. An example callback, being on a Period with @id value of 0, using a dummy URL and selecting the period labeled as “blue” would be:

http://cdn.com/content\_xyz/0/selection?parameter=blue

For an application that wants to only show the textual description of each choice on screen, the carriage of that information in Event body or @messageData, and @data, makes the logic of the application straightforward.

* + 1. Graph Update

The server upon the reception of the callback request parses the parameters and creates a new MPD version, extended with the adequate next Period to reflect the client selection. The client requests the new MPD version according to the minimum update period or based on the MPD validity expiration event. If the new Period is not an end edge, a further Event element in the current Period will trigger the callback of the client and will make the process continue until the DASH Client consumes a Period without an Event in which case no further MPD version is created and the end of the presentation is reached by the DASH Client.

*Add to Annex G after G.25:*

G.26 Nonlinear playback (Annex L)

An example EventStream (with an example non-MPEG defined value attribute), containing an example Event element and the respective CallbackInfo elements and of nonlinear playback event at P0 Period in Figure XXX is shown below:

|  |
| --- |
| <EventStream schemeIdUri="urn:mpeg:dash:nonlinearplayback:2020" value="urn:xapp:2020:userinterface1" >      <Event  presentationTime="PT530S"  duration= "30"  nlp:selectionInfo = "What do you like to happen next?" contactURL="http://cdn.com/content\_xyz/">  <SelectionInfo parameter="1" data="Bill kills Alice">          <SelectionInfo parameter="2" data="Bill kisses Alice">          <SelectionInfo parameter="3" data="Bill frames Alice">          <SelectionInfo parameter="blue" data="Bill kisses Alice">          <SelectionInfo parameter="red" data="Bill frames Alice">          <SelectionInfo parameter="default">      </Event>  </EventStream> |

**Figure XXX. EventStream for Figure L.1 example**

Note that in the above example there are six CallbackInfo elements, even though there are three choices since a path choice can be expressed in more than one way.

*Add to 3.2 Symbols and abbreviated terms*

|  |  |
| --- | --- |
| ARI | Addressable resource index |

*Add a new Annex M*

**Annex M**

**(normative)**

**Addressable Resource Index Track**

* 1. Motivation and High-level Solution

The following aspects are observed

* In several cases there is a desire that an adaptive streaming client has exact knowledge of the duration and size of addressable resources and possible a subset of those on the server.
* Addressable Resources are Track Files, Segments or Chunks in the CMAF context, but apply equally to DASH or HLS.
* For on-demand services, an exact map of this information may be provided by the Segment Index.

However, there are cases for which additional information on segment information may be beneficial for the client and possibly network operation, for which the Segment Index is not sufficient. Examples include:

* A solution is required for different operation modes: low-latency live, live, time-shifted, VoD
* The solution is expected to work for different target latency of the client
* The client and network address to operate in different network conditions
* The message also includes information on the content quality

NOTE: Even though this track uses CMAF terminology, this can be applied to DASH Adaptation Sets that are not conforming to CMAF.

The Addressable Resource Index (ARI) Track provides a solution to the above use cases by describing all details of the Addressable resources and sub-sets of a CMAF Switching Set in a metadata track.

An ARI Track is applied to a CMAF Switching Set for which each CMAF Track has identical Segment, Fragment and Chunk Structure in terms of duration. The following principles apply:

* The ARI Track is time-aligned with the CMAF Switching Set.
* The ARI Track documents the properties of several or all tracks of the CMAF Switching Set
* A Header information is defined for the metadata track
* A sample of the ARI track is associated to each CMAF chunk.
* The sample contains detailed information of the time-aligned CMAF chunks in several or all CMAF Tracks of the CMAF Switching Set.
* Each sample of the ARI Track is a sync sample.

Delivery and Segmentation of the track is independent of the Chunk/Segment Structure of the associated switching set.

The ARI track provide a solution that can be used with live or on-demand presentations, including live presentations that are converted to on-demand after the end of the live session.

* 1. Definition: CMAF Addressable Resource Index
     1. Definition

Sample Entry Type: 'cari'   
Container: Sample Description Box ('stsd')  
Mandatory: No  
Quantity: 0 or 1

This metadata describes all details of the addressable resources and sub-sets of a CMAF Switching Set as defined in ISO/IEC 23000-19 in a metadata track.

It is assumed that

* for several or all Tracks in the CMAF Switching Sets the same Segment, Fragment and Chunk structure applies.
* Each of the CMAF tracks can be uniquely identified by a track\_id.

The following principles are applied:

* The ARI Track is time-aligned with the tracks of the CMAF Switching Set.
* The ARI Track documents the properties of all tracks of the CMAF Switching Set
* A Header information is defined for the metadata track
* A sample of the track is defined for each CMAF chunk in a time-aligned manner. The association of the CMAF chunk and the metadata sample is done such that the baseMediaDecodeTime of the CMAF chunk is identical to the sample time in the metadata track.
* The sample contains detailed information for the CMAF chunk in each of the tracks in the switching set
* Note that this track may even be used to carry for example Events or Producer Reference time for the Media Presentation.
  + 1. Syntax

CMAF Addressable Resource Index Metadata use the following sample entry:

class CmafAriMetaDataSampleEntry() extends MetaDataSampleEntry ('carc') {  
 CmafAriConfigurationBox();  
}

aligned(8) class CmafAriConfigurationBox extends FullBox('carc', version = 0, flags = 0) {  
 unsigned int(32) switching\_set\_identifier;  
 unsigned int(10) num\_tracks;  
 unsigned int(10) num\_quality\_indicators;  
 unsigned int(1) edrap\_allowed\_flag;  
 bit(11) reserved;  
 for(i=1; i <= num\_tracks; i++)  
 unsigned int(32) track\_id;  
 // provides the order of the tracks for each sample  
 // additional information on the CMAF Switching Set may be provided  
 for(i=1; i <= num\_quality\_indicators; i++)  
 string quality\_identifier;  
}

CMAF Addressable Resource Index samples use the following syntax:

class CmafAriFormatStruct () {  
 for(i=1; i <= num\_tracks; i++) {  
 unsigned int(1) segment\_start\_flag;  
 unsigned int(1) marker;  
 unsigned int(3) SAP\_type;  
 unsigned int(1) emsg\_flag;  
 unsigned int(1) prft\_flag;  
 unsigned int(1) sap\_is\_edrap\_flag;  
 bit(24) reserved;  
 unsigned int(32) offset  
 unsigned int(32) size;  
 for(i=1; i <= num\_quality\_indicators; i++){  
 unsigned int(32) quality;  
 }  
 unsigned int(1) loss;  
 bit(15) reserved;  
 unsigned int(8) num\_prediction\_pairs;  
 for(i=1; i <= num\_prediction\_pairs; i++) {  
 unsigned int(32) prediction\_min\_window;  
 unsigned int(32) predicted\_max\_bitrate;  
 }

}  
}

* + 1. Semantics

switching\_set\_identifier specifies a unique identifier for the switching set in the context of the application.

num\_tracks indicates the number of tracks indexed in the ARI track.

num\_quality\_indicators specifies the number of quality indicators used for identifying the quality of the CMAF chunk.

edrap\_allowed\_flag specifies whether extended dependent random access point (EDRAP) samples may be present in one or more of the tracks indexed in the ARI track. An EDRAP sample is a sample for which all subsequent samples in the same track in both decoding and output order can be correctly decoded provided that the closest preceding SAP sample of type 1, 2, or 3 and zero or more preceding EDRAP samples are available when decoding the sample and the subsequent samples.

quality\_identifier specifies an identifier that tells how the quality values in the sample are expected to be interpreted. This is a 4CC code that can be registered.

track\_ID provides the selection and ordering in the samples of the tracks using the track\_IDs.

segment\_start\_flag indicates whether the CMAF chunk is the start of a segment.

marker identifies if this CMAF chunk includes at least one styp box.

SAP\_type, when greater than 0, identifies the SAP type of the sample this CMAF chunk starts with. The semantics of SAP\_type equal to 0 is unspecified..

emsg\_flag indicates whether this CMAF chunk provides at least one emsg box.

prft\_flag indicates whether this CMAF chunk includes at least one prft box.

sap\_is\_edrap\_flag indicates whether the SAP this CMAF chunk starts with is an EDRAP. When edrap\_allowed\_flag equal to 0, the value of sap\_is\_edrap\_flag shall be equal to 0.

offset identifies the offset of the CMAF chunk from the start of the segment.

size provides the size in octets of the CMAF chunk.

quality provides the quality of the CMAF chunk according to a given quality scheme identifier. The data type of the quality value (integer or float) is defined by the quality scheme. If the quality scheme identifier is a null string, then quality is an unsigned integer, interpreted linearly with quality increase with increasing value.

loss indicates that the media data of the CMAF chunk is lost.

num\_prediction\_pairs provides how many pairs of the expected prediction values are provided.

prediction\_min\_windows provides a value for minbuffer time identical to the MPD value.

predicted\_max\_bitrate provides a value for bandwidth identical to the MPD semantics that holds for the duration of the prediction\_min\_windows value.

* 1. DASH Media Presentation addition and operation

In the MPD, the following signalling needs to be done:

1. The metadata track is provided as a regular Adaptation Set with a single track
2. A Switching Set is associated with this track
3. The streaming is done as regular, but any optimizations can be done:
   1. Availability time offset
   2. Chunking
   3. Segmentation

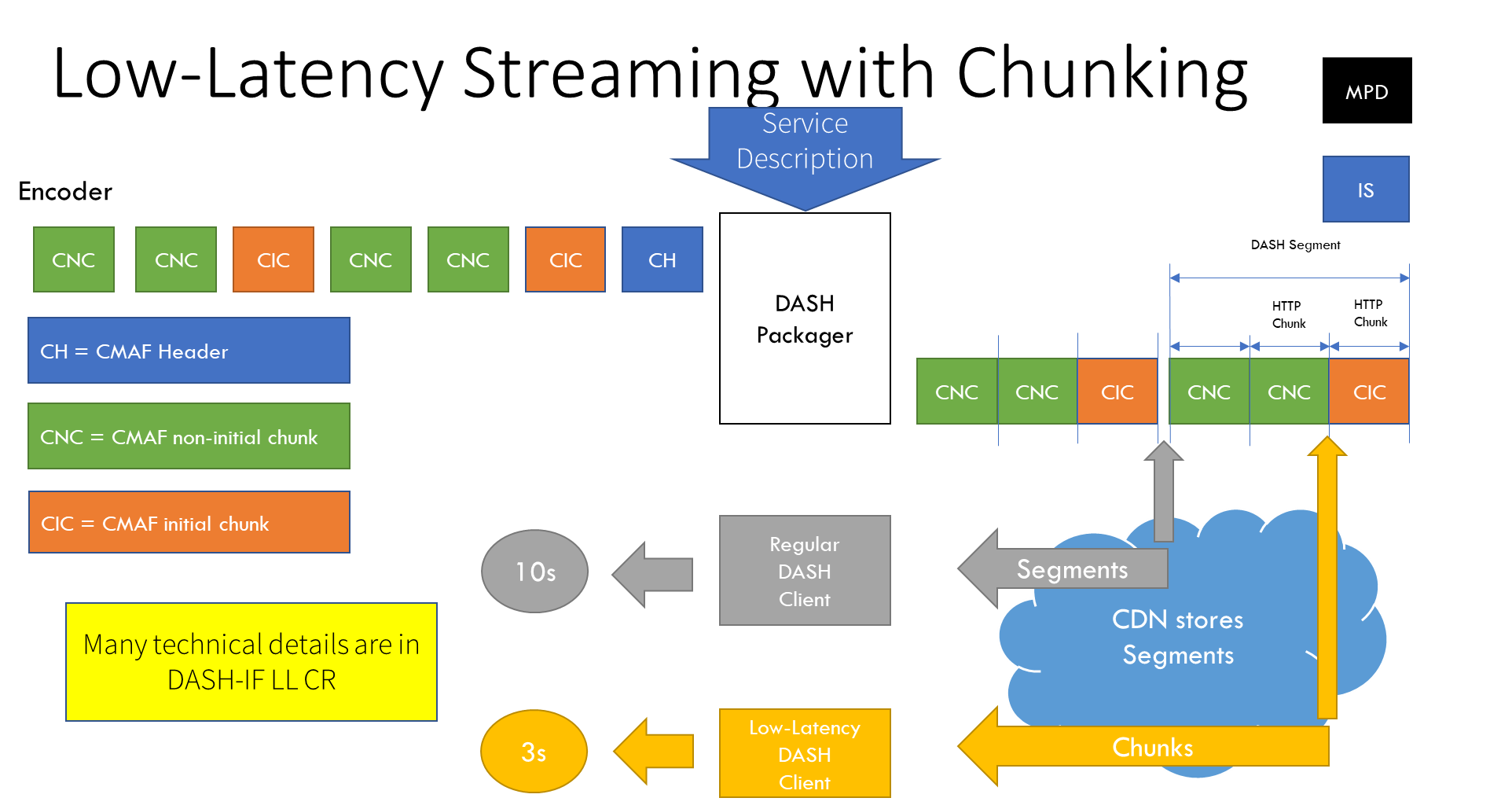
It is beneficial if the metadata track can be accessed ahead or at least together with the segment availability times.

The DASH client implements the metadata processor of this track and makes use of the information.

This really simplifies the overall addition, as all existing streaming technologies can be applied.

* 1. Illustration

Assuming a low-latency streaming as shown in Figure M-1.



**Figure M-1. Low-latency streaming with chunking**

The DASH packager can create all the metadata and add this to stream an ARI Metadata track as shown below. Every set of CMAF chunks at the same media time results in a sample in the metadata track.

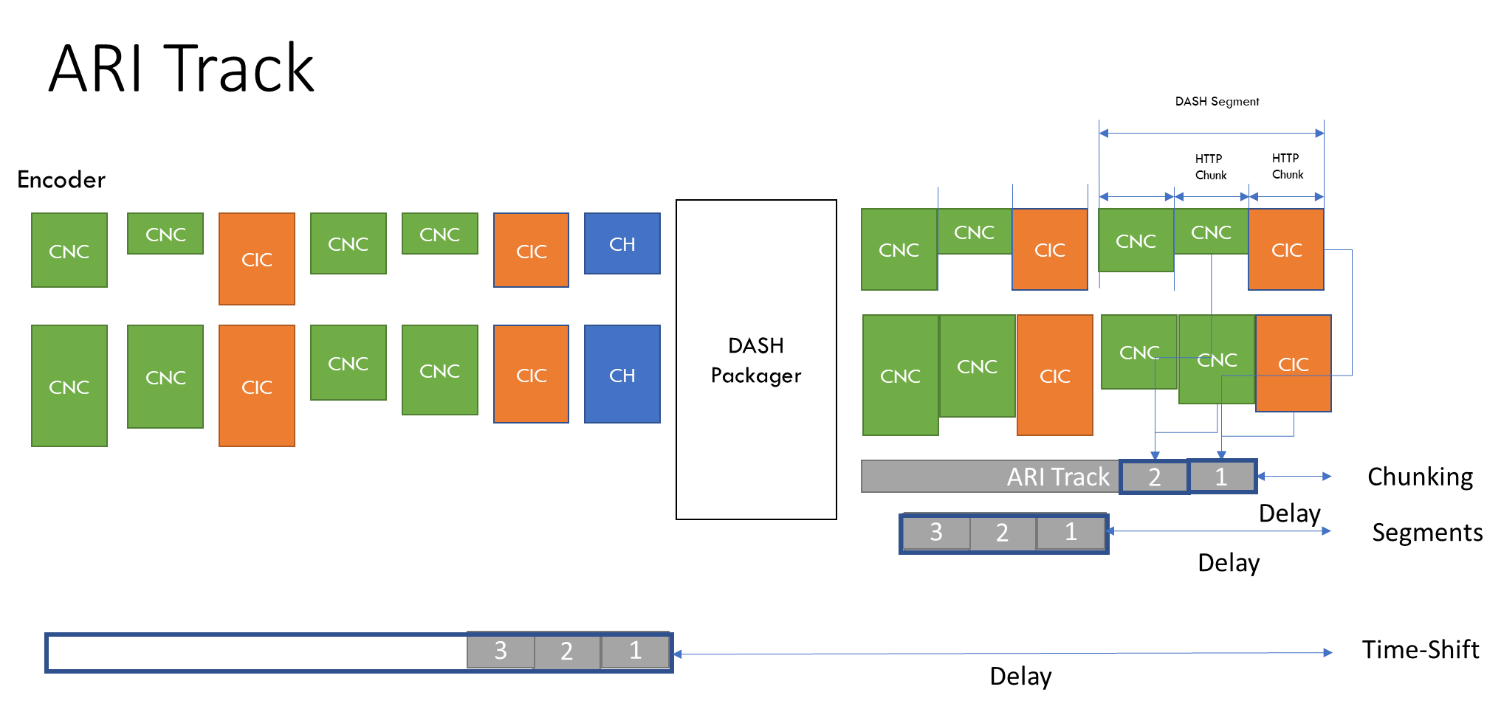
The interesting aspect is, that the publication delay of the metadata can be done flexibly and is then a matter of regular streaming optimization aspects.

* More or less segments
* More or less requests
* More or less chunks
* Scalability and so on

The information can also be used in exactly the same manner for live, on-demand and time-shifted streaming.

Content can be removed as well based on a customized time-shift buffer.

Figure M-2 shows the basic concept that there is flexibility on how to design the ARI Track (chunked, segmented, or long segments). The tradeoffs are latency and delay for the live service that one can use this information and the possible overhead.



**Figure M-2. ARI Track flexibility/trade-offs for segment and chunk delivery**

Additional information may be added to the metadata track.

* 1. Quality definitions

Table M-1 defines different quality identifiers that may be used in combination with the ARI Track quality\_identifier as well as a definition of the value for the quality.

**Table M-1— Quality identifiers and definitions**

|  |  |
| --- | --- |
| Identifier | Definition |
| 'qlin' | A basic quality identifier indicating that the quality assigned to the CMAF chunk is measured in linear scale independent of the output environment. The quality is measured in linear scale, i.e. a factor of 2 increases the perceived quality by a factor of 2 based on the content authors judgement. |
| 'q720' | A quality identifier indicating that the quality assigned to the CMAF chunk is measured if the CMAF chunk is displayed as 720p output resolution. The quality is measured in linear scale, i.e. a factor of 2 increases the perceived quality by a factor of 2 based on the content authors judgement. |
| 'q\_2k' | A quality identifier indicating that the quality assigned to the CMAF chunk is measured if the CMAF chunk is displayed as 1080p output resolution. The quality is measured in linear scale, i.e. a factor of 2 increases the perceived quality by a factor of 2 based on the content authors judgement. |
| 'q\_4k' | A quality identifier indicating that the quality assigned to the CMAF chunk is measured if the CMAF chunk is displayed as 2160p output resolution. The quality is measured in linear scale, i.e. a factor of 2 increases the perceived quality by a factor of 2 based on the content authors judgement. |
| 'q\_8k' | A quality identifier indicating that the quality assigned to the CMAF chunk is measured if the CMAF chunk is displayed as 4320p output resolution. The quality is measured in linear scale, i.e. a factor of 2 increases the perceived quality by a factor of 2 based on the content authors judgement. |

*Change Figure A.13.1 with:*

Manifest Parser

Timed Metadata Track Parser

Media Buffer

Media Decoder

Event/Metadata Synchronizer & Dispatcher

Inband Event & ‘moof’ Parser

HTTP Stack

Event/T. Metadata dataflow

Media dataflow

Control/Synchronization

DASH Client’s Control, Selection & Heuristic Logic

Application

DASH

Access API

MPD

Events

Inband

Events

Timed

Metadata

DASH Events & Metadata

Event/

Metadata

API

Subscribe

**Media Segments**

API

Event and Timed Metadata Buffer

Current Presentation Time

**Figure A.13-1- DASH Player architecture including the Event Stream timed metadata track handling**

*Change item 7.b of A.13.2:*

1. For on-start dispatch mode, dispatch the message data of the event their associated presentation time or latest before the event duration has ceased, or timed metadata samples at their presentation time using the synchronization signal from the media decoder.

*with:*

1. For on-start dispatch mode, dispatch the message data of the event their associated presentation time or latest before the event duration has ceased, or timed metadata samples at their presentation time using the synchronization signal from the media decoder. For synchronized dispatch between media presentation and event dispatch, the media decoder and render provides the current presentation time to the Event & Metadata Synchronizer and Dispatcher function.

*Change A.13.5:*

A.13.5 MPD Events timing model

MPD Events follow an equivalent data model to inband Events but are carried in the MPD within a **Period** element. Each Period event can have one or multiple **EventStream** elements, defining the **EventStream**@schemeIdUri, **EventStream**@value, **EventStream**@timescale, and contained sequences of Event elements. Each event may have **Event**@presentationTime, **Event**@duration, **Event**@id, and **Event**@messageData attributes as specified in 5.10.2. As is shown in Figure A.3, each MPD Event has three timing parameters along the media timeline:

1. The Latest Arrival Time *(LAT)* which is one of the following values:
   1. PeriodStart of the Period element containing the Event if the Period has not been played yet
   2. The moment of media timeline when the period is random accessed for the first time
   3. The moment in which the Period element of an MPD update is parsed and this event is added by the update while the period is being played.
2. Event Start Time (ST): the moment in the media timeline that a given MPD Event becomes active and can be calculated from the **attributeEvent**@presentationTime.
3. Event duration (DU): the duration for which the event is active that can be calculated from the attribute **Event**@duration.

Note that the first parameter is inherited from the Period containing the Events and only the 2nd and 3rd parameters are explicitly included in the Event element. Each **EventStream** also has **EventStream**@timescale to scale the above parameters.

Figure A.3 demonstrates these parameters in the media timeline.

Latest Arrival Time

(LAT)

Event Duration (DU)

P(n-1)

Media timeline

Periods

P(n+1)

Event Start Time (ST)

P(n)

1. MPD events timing model

The ST of an MPD event, relative to period start time of Period containing the Event, can be calculated using values in its **EventStream** and **Event** elements:

where:

* *PST* is the Period start time of the Period containing the event stream.
* *PTO* is the presentation time offset of the Event Stream provided by **EventStream**@presentationTimeoffset.
* *TS* is the value of **EventStream**@timescale.
* *PT* is the value of **EventStream**@presentationTime.

In this Annex, we use the following common variable names instead of some of above variables to harmonize parameters between Inband events, MPD events, and timed metadata samples:

* scheme\_id = **EventStream**@schemeIdUri
* value = **EventStream**@value
* presentation\_time = ST
* duration = **Event**@duration/**EventStream**@timescale
* id = **Event**@id
* message\_data[] = decode64(**Event**@messageData)

in which decode() function is:

Note that the DASH client is expected to Base64 decode the **Event**@messageData value if the received **Event**@contentEncoding value is base64.

Note that the Event duration may be unknown.

*Change A.13.8:*

It is assumed that the application is subscribed to a specific event stream identified by a (scheme/value) pair with a specific dispatch\_mode, either ***on-start*** or ***on-receive***, as described in subclause A.13.7.

The processing model varies depending on *dispatch\_mode*.

1. Common process
   1. The DASH Player implements the following process:
      1. Parse the 'emsg'/timed metadata sample and retrieve scheme\_uri/(value).
      2. If Application is not subscribed to the scheme\_uri/(value) pair, end the processing of this 'emsg'.
2. on-receive processing
   1. The DASH Player implements the following process when *dispatch\_mode* = *on\_receive*:
      1. Dispatch the event/timed metadata, including *ST*, *id*, *DU*, *timescale*, and *message\_data* as described in subclause A.13.6.
3. on-start processing
   1. The DASH Player sets up an Active Event Table for each subscribed *scheme\_uri*/(*value*) in the case of *dispatch\_mode* = *on\_start*. The ***Active Event Table*** maintains a single list of 'emsg' *ids* that have been dispatched.
   2. The DASH Player implements the following process when *dispatch\_mode* = *on\_start*:
      1. Derive the event instance/metadata sample’s *ST*
      2. If the current media presentation time value is smaller than *ST*, then go to step v.
      3. Derive the ending time *ET*= *ST* + *DU*.
      4. If the current presentation time value is greater than *ET*, then end processing.
      5. In the case of event: Compare the event’s *id* with the entries of the Active Event Table of the same *scheme\_uri*/(*value)* pair:
         * If an entry with the identical *id* value exists, end processing;
         * If not, add 'emsg'’s *id* to the corresponding the [Active Event Table](file:///C:\Users\irajs\Downloads\23009-1-5th-edition-MPEG133-2021-04-28-iraj.docx#active-event-table).
      6. Dispatch the event/metadata message\_data at time ST, or immediately if the current presentation time is larger than ST, as described in subclause A.13.6.

*with:*

It is assumed that the application is subscribed to a specific event stream identified by a (scheme/value) pair with a specific dispatch\_mode, either ***on-start*** or ***on-receive***, as described in subclause A.13.7.

The processing model varies depending on *dispatch\_mode*.

1. Common process
   1. The DASH Client implements the following process:
      1. The DASH Client sets up an ***Pending*** Event Table ***(PET)*** for each subscribed *scheme\_uri*/(*value*) in the case of *dispatch\_mode* = *on\_start*. The PET maintains a single list of event *ids* that are waiting to be dispatched. The DASH Client also sets up an ***Dispatched Event Table*** ***(DET)*** for each subscribed *scheme\_uri*/(*value*). TheDET maintains a single list of 'emsg' *ids* that have been dispatched.
      2. Parse the 'emsg'/timed metadata sample and retrieve scheme\_uri/(value).
      3. If Application is not subscribed to the scheme\_uri/(value) pair, end the processing of this 'emsg'.
      4. Derive the event instance/metadata sample’s *ST*
      5. Derive the ending time *ET*= *ST* + *DU*.
2. on-receive processing
   1. The DASH Client implements the following process when *dispatch\_mode* = *on\_receive*:
      1. If the current presentation time value is greater than *ET*, then end processing.
      2. In the case of event: Compare the event’s *id* with the entries of the DET of the same *scheme\_uri*/(*value)* pair:
         * If an entry with the identical *id* value exists, end processing;
      3. Dispatch the event/timed metadata, including *ST*, *id*, *DU*, *timescale*, and *message\_data* as described in subclause A.13.6, and add the event to the DET.
3. on-start processing
   1. DASH ClientThe DASH Client implements the following process when *dispatch\_mode* = *on\_start*:
      1. Derive the event instance/metadata sample’s *ST*
      2. If the current media presentation time value is smaller than *ST*, then go to step v.
      3. Derive the ending time *ET*= *ST* + *DU*.
      4. If the current presentation time value is greater than *ET*, then end processing.
      5. In the case of event: Compare the event’s *id* with the entries of the PET and DET of the same *scheme\_uri*/(*value)* pair:
         * If an entry with the identical *id* value exists in either table, end processing;
         * If not, add 'emsg'’s *id* to the corresponding [PET](#active-event-table).
      6. Dispatch the event/metadata message\_data at time ST, or immediately if the current presentation time is larger than ST, as described in subclause A.13.6, remove the event, if exists, from the PET and add it to DET.

*Change A.13.9:*

Along with the media samples, the event instances and timed metadata samples are buffered. The event/metadata buffer should be managed with the same scheme as the media buffer, i.e. as long as a media sample exists in the media buffer, the corresponding events and/or metadata sample are maintained in the event/metadata buffer.

*with:*

Along with the media samples, the event instances and timed metadata samples are buffered. The event/metadata buffer should be managed with the same scheme as the media buffer, i.e. as long as a media sample exists in the media buffer, the corresponding events (the inband events that are carried by the segment containing the media sample/the MPD events that are included in the Period element containing the media sample) and the aligned metadata samples with the media sample are maintained in the event/metadata buffer.

*Change A.13.10 to:*

**A.13.10 Prose description of APIs**

The event/timed metadata API is an interface defined between a DASH client and a device application in the exchange of subscription data and dispatch/transfer of matching DASH Event or timed metadata information between these entities. The Event/timed metadata API is shown in Figure A.1.

NOTE 1 In this Annex, the term "DASH Player" is used.

The description of the API below is strictly functional, i.e. implementation-agnostic. For example, the subscribeEvent() method as defined below may be mapped to the existing on(type,listener,scope) method as defined for the dash.js under MediaPlayerEvents.

As part of this API and before any operations, the DASH Player provides a list of *scheme\_id*/(*value*) listed in the MPD when it receives it. This list includes all MPD and inband events as well as *scheme\_id* of all timed metadata tracks. At this point, the Application is aware of the possible events and metadata deliverable by the DASH Player.

NOTE 2 The DASH Player may provide the Application the list of DASH event schemes as a part of listed event schemes in the MPD and consequently, the Application may subscribe to one or more of these event schemes.

*Change A.13.12 to the following:*

**A.13.12 Dispatch modes for DASH-specific events**

In 5.10.4, several DASH-specific event schemes are defined. Table A.4 describes their dispatch modes.

Table A.4 DASH-specific event schemes dispatch modes

| **scheme** | **Dispatch mode** |
| --- | --- |
| urn:mpeg:dash:event:2012 | on-receive |
| urn:mpeg:dash:event:callback:2015 | on-start |
| urn:mpeg:dash:event:ttfn:2016 | on-start |
| urn:mpeg:dash:event:period:2020 | On-receive |

*Change Table A.1-1 with:*

Table A.13-1. Event/timed metadata API parameters and datatypes

| **API Parameter** | **MPD event** | **Inband 'emsg'** | **Metadata** | **Data Type** | **‘on-receive’** | **‘on-start’** |
| --- | --- | --- | --- | --- | --- | --- |
| scheme\_id | **EventStream**@schemeIdUri | scheme\_id\_uri | timed metadata track URI | string | Y | Y |
| value | **EventStream**@value | value |  | string | Y | Y |
| presentation\_time | **Event**@presentationTime/ **EventStream**@timescale | presentation\_time/timescale | timed metadata sample presentation time – presentation time offset | double(64)  seconds | Y | N |
| duration | **Event@**duration/ **EventStream@**timescale | duration/timescale | timed metadata sample duration | double(64)  in seconds | Y | N |
| id | **Event**@id | id |  | unsigned int(32) | Y | N |
| message\_data | **Event**@messageData | message\_data[] | timed metadata sample data in mdat | unsigned int(8) x messageSize | Y | Y |
| Y= Yes, N= NO, O= Optional | | | | | | |

*Change cross A.13:*

DASH Player

*with*

DASH Client

*Change Table I.3 to:*

Table I.3 — Semantics of ExtendedUrlInfoType element

| **Element or Attribute Name** | | | **Use** | **Description** |
| --- | --- | --- | --- | --- |
|  | ExtendedUrlInfoType | |  | provides information for derivation of parameter string. This is an extension of UrlQueryInfoType element defined in Table I.1. |
|  |  | @includeInRequests | OD  (default: "segment") | specifies which HTTP GET requests shall carry parameters. Value is a white spaced concatenated list of the following keys:  1) "segment" (all segment requests)",  2) "xlink" (all XLink resolution requests),  3) "mpd" (all MPD requests),  4) "callback" (all requests triggered by DASH callback events),  5) "chaining" (requests for chained-to MPDs,  6) "fallback" (requests for the alternative MPDs.  7) “sbd” (requests for the SBD document of ISO/IEC 23009-8)Default value is "segment", i.e. parameters will be only sent with segment requests  NOTE   Depending on the actual element used, parameter output goes either to query parameters (for ExtUrlQueryInfo) or HTTP headers (for ExtHttpHeaderInfo) |
|  |  | @headerParamSource | OD  (default: "segment") | specifies HTTP responses from which HTTP header values, identified by the template $*header:header-name*$, should be extracted from. Value is a white spaced concatenated list of the following keys :  1) "segment" (all segment requests)",  2) "xlink" (all XLink resolution requests),  3) "mpd" (all MPD requests),  4) "callback" (all requests triggered by DASH callback events).  5) “sbd” (requests for the SBD document of ISO/IEC 23009-8)Default value: empty string (no header parameters inspected)  If this attribute is present then: (a) @queryTemplate attribute shall be present and shall contain the $header:<header-name>$ identifier, and (b) neither @useMPDUrlQuery nor @queryString attribute shall be present. |
|  |  | @sameOriginOnly | OD | specifies that parameters must only be sent to the same origin they were instantiated from. In case of HTTP headers as source, the origin is defined as the origin of the HTTP request identified by the attribute @headerParamSource. In case the parameters are instantiated from the MPD or from the MPD URL, the origin is defined in both case by the MPD URL.  Two origins are the same as defined by IETF RFC 6454, i.e. same scheme/host/port triple (see 5. Comparing Origins)  Default value: false (no origin restrictions) |
| **Key**  For attributes: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory.  For elements: <minOccurs>...<maxOccurs> (N=unbounded)  The conditions only hold without using xlink:href. If linking is used, then all attributes are "optional" and <minOccurs=0>.  Elements are **bold**; attributes are non-bold and preceded with an @. | | | | |

The following identifiers are defined, in addition to the ones defined in Table I.2

*Add to Annex G:*

**G.27 Multi-key encryption**

In many cases content rights agreements stipulate different requirements for different classes of premium content. For example, software decoding and decryption are often sufficient for content with 576 or less horizontal lines, while hardware decryption and decoding are required for UltraHD content. Similarly, different degrees of output protection are needed for different resolutions.

The example below shows an MPD for a content offering with three classes of video content (SD, HD, and UHD) with different keys for each class. Each class of resolutions corresponds to a single adaptation set, and adaptation set switching (see 5.3.3.5 above) is used to switch between them.

Depending on the content and output protection features detected by the DRM client, the client is given access to all or some keys. The MPD below also states the DRM robustness and output protection characteristics needed for playback. The client is expected to use these to avoid switching into representations it will not be allowed to present.

NOTE 1: robustness values are DRM-specific

NOTE 2: The license server response is the source of truth in case of a mismatch between the capability detection done by the client and by the DRM agent.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <MPD xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:cenc="urn:mpeg:cenc:2013" type="dynamic" id="42" profiles="urn:mpeg:dash:profile:isoff-live:2011" minBufferTime="PT2.000S" maxSegmentDuration="PT0H0M2.016S" minimumUpdatePeriod="PT0H0M2.002S" availabilityStartTime="1977-05-25T18:00:00.000Z" timeShiftBufferDepth="PT0H0M30.000S" publishTime="2021-04-17T04:15:27.145Z">  <Period id="807136760" start="PT384015H43M16.234S">  <!-- SD Adaptation Set (keyID: ed1f2e89-8a1f-47f8-a5f5-371dd397464c) -->  <AdaptationSet id="sd-video" contentType="video" mimeType="video/mp4" segmentAlignment="true" startWithSAP="1">  <ContentProtection schemeIdUri="urn:mpeg:dash:mp4protection:2011" value="cenc" cenc:default\_KID="ed1f2e89-8a1f-47f8-a5f5-371dd397464"/>    <ContentProtection schemeIdUri="urn:uuid:afbcb50e-bf74-3d13-be8f-13930c783962" robustness="SW\_SECURE\_DECODE">  <cenc:pssh>...</cenc:pssh>  </ContentProtection>    <ContentProtection schemeIdUri="urn:uuid:9a04f079-9840-4286-ab92-e65be0885f95" value="MSPR 2.0" robustness="SL2000">  <cenc:pssh>...</cenc:pssh>  </ContentProtection>  <!-- Can switch to HD or UHD -->  <SupplementalProperty schemeIdUri="urn:mpeg:dash:adaptation-set-switching:2016" value="hd-video,uhd-video" />  <Accessibility schemeIdUri="urn:scte:dash:cc:cea-608:2015" value="CC1=eng"/>  <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>    <SegmentTemplate initialization="$RepresentationID$/init.mp4" media="$RepresentationID$/$Time$.mp4" timescale="90000" startNumber="807170070" presentationTimeOffset="36403">  <SegmentTimeline>  <S t="6002913283" d="180180" r="13"/>  </SegmentTimeline>  </SegmentTemplate>  <Representation id="root\_video4" bandwidth="769600" codecs="hvc1.2.4.L93.B0" width="512" height="288" frameRate="30000/1001"/>  </AdaptationSet>  <!-- HD Adaptation Set (keyID: 65ee94f8-54db-4460-ae6d-401bf195fc2b) -->  <AdaptationSet id="hd-video" contentType="video" mimeType="video/mp4" segmentAlignment="true" startWithSAP="1">  <ContentProtection schemeIdUri="urn:mpeg:dash:mp4protection:2011" value="cenc" cenc:default\_KID="65ee94f8-54db-4460-ae6d-401bf195fc2b"/>    <ContentProtection schemeIdUri="urn:uuid:afbcb50e-bf74-3d13-be8f-13930c783962" robustness="HW\_SECURE\_CRYPTO">  <cenc:pssh>...</cenc:pssh>  </ContentProtection>    <ContentProtection schemeIdUri="urn:uuid:9a04f079-9840-4286-ab92-e65be0885f95" value="MSPR 2.0" robustness="SL2000">  <cenc:pssh>...</cenc:pssh>  </ContentProtection>  <!-- Can switch to SD or UHD -->  <SupplementalProperty schemeIdUri="urn:mpeg:dash:adaptation-set-switching:2016" value="sd-video,hd-video" />  <Accessibility schemeIdUri="urn:scte:dash:cc:cea-608:2015" value="CC1=eng"/>  <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>  <SegmentTemplate initialization="$RepresentationID$/init.mp4" media="$RepresentationID$/$Time$.mp4" timescale="90000" startNumber="807170070" presentationTimeOffset="36403">  <SegmentTimeline>  <S t="6002913283" d="180180" r="13"/>  </SegmentTimeline>  </SegmentTemplate>  <Representation id="root\_video3" bandwidth="2282000" codecs="hvc1.2.4.L120.B0" width="1280" height="720" frameRate="30000/1001"/>  <Representation id="root\_video2" bandwidth="7088800" codecs="hvc1.2.4.L123.B0" width="1920" height="1080" frameRate="30000/1001"/>  <Representation id="root\_video1" bandwidth="7088800" codecs="hvc1.2.4.L123.B0" width="1920" height="1080" frameRate="60000/1001"/>  </AdaptationSet>  <!-- UHD Adaptation Set (keyID: 100efe5e-247f-4a82-b4ed-cb159f8b7b20) -->  <AdaptationSet id="uhd-video" contentType="video" mimeType="video/mp4" segmentAlignment="true" startWithSAP="1">  <ContentProtection schemeIdUri="urn:mpeg:dash:mp4protection:2011" value="cenc" cenc:default\_KID="100efe5e-247f-4a82-b4ed-cb159f8b7b20"/>  <ContentProtection schemeIdUri="urn:uuid:afbcb50e-bf74-3d13-be8f-13930c783962" robustness="HW\_SECURE\_ALL">  <cenc:pssh>...</cenc:pssh>  </ContentProtection>  <ContentProtection schemeIdUri="urn:uuid:9a04f079-9840-4286-ab92-e65be0885f95" value="MSPR 2.0" robustness="SL3000">  <cenc:pssh>...</cenc:pssh>  </ContentProtection>  <!-- Can switch to SD or HD -->  <SupplementalProperty schemeIdUri="urn:mpeg:dash:adaptation-set-switching:2016" value="sd-video,hd-video" />  <!-- HDCP Hints -->  <OutputProtection schemeIdUri="urn:mpeg:dash:output-protection:hdcp:2020" value="2.2"/>  <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>  <SegmentTemplate initialization="$RepresentationID$/init.mp4" media="$RepresentationID$/$Time$.mp4" timescale="90000" startNumber="807170070" presentationTimeOffset="36403">  <SegmentTimeline>  <S t="6002913283" d="180180" r="13"/>  </SegmentTimeline>  </SegmentTemplate>  <Representation id="root\_video1" bandwidth="14057200" codecs="hvc1.2.4.L153.B0" width="2560" height="1440" frameRate="60000/1001"/>  <Representation id="root\_video0" bandwidth="20575600" codecs="hvc1.2.4.L153.B0" width="3840" height="2160" frameRate="60000/1001"/>  </AdaptationSet>  <!-- E-AC-3 Audio Primary lang (keyID: 55ff04db-d75a-dfb9-ef8a-6473ae9ac9c4) -->  <AdaptationSet id="3" contentType="audio" mimeType="audio/mp4" lang="en">  <AudioChannelConfiguration schemeIdUri="urn:mpeg:mpegB:cicp:ChannelConfiguration" value="6"/>  <ContentProtection schemeIdUri="urn:mpeg:dash:mp4protection:2011" value="cenc" cenc:default\_KID="55ff04db-d75a-dfb9-ef8a-6473ae9ac9c4"/>  <ContentProtection schemeIdUri="urn:uuid:afbcb50e-bf74-3d13-be8f-13930c783962" robustness="SW\_SECURE\_CRYPTO">  <cenc:pssh>...</cenc:pssh>  </ContentProtection>  <ContentProtection schemeIdUri="urn:uuid:9a04f079-9840-4286-ab92-e65be0885f95" value="MSPR 2.0" robustness="SL2000">  <cenc:pssh>...</cenc:pssh>  </ContentProtection>  <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>  <SegmentTemplate initialization="$RepresentationID$/init.mp4" media="$RepresentationID$/$Time$.mp4" timescale="90000" startNumber="807170070" presentationTimeOffset="36403">  <SegmentTimeline>  <S t="6002915623" d="178560" r="0"/>  <S t="6003094183" d="181440" r="0"/>  <S t="6003275623" d="178560" r="0"/>  <S t="6003454183" d="181440" r="0"/>  <S t="6003635623" d="178560" r="0"/>  <S t="6003814183" d="181440" r="1"/>  <S t="6004177063" d="178560" r="0"/>  <S t="6004355623" d="181440" r="0"/>  <S t="6004537063" d="178560" r="0"/>  <S t="6004715623" d="181440" r="0"/>  <S t="6004897063" d="178560" r="0"/>  <S t="6005075623" d="181440" r="1"/>  </SegmentTimeline>  </SegmentTemplate>  <Representation id="root\_audio66" bandwidth="288000" codecs="ec-3" audioSamplingRate="48000"/>  </AdaptationSet>  <!-- Same as previous audio -->  <AdaptationSet id="4" contentType="audio" mimeType="audio/mp4" lang="en">  <AudioChannelConfiguration schemeIdUri="urn:mpeg:mpegB:cicp:ChannelConfiguration" value="2"/>  <ContentProtection schemeIdUri="urn:mpeg:dash:mp4protection:2011" value="cenc" cenc:default\_KID="55ff04db-d75a-dfb9-ef8a-6473ae9ac9c4"/>  <ContentProtection schemeIdUri="urn:uuid:afbcb50e-bf74-3d13-be8f-13930c783962" robustness="SW\_SECURE\_CRYPTO">  <cenc:pssh>...</cenc:pssh>  </ContentProtection>  <ContentProtection schemeIdUri="urn:uuid:9a04f079-9840-4286-ab92-e65be0885f95" value="MSPR 2.0" robustness="SL2000">  <cenc:pssh>...</cenc:pssh>  </ContentProtection>  <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>  <SegmentTemplate initialization="$RepresentationID$/init.mp4" media="$RepresentationID$/$Time$.mp4" timescale="90000" startNumber="807170070" presentationTimeOffset="36403">  <SegmentTimeline>  <S t="6002916699" d="180480" r="0"/>  <S t="6003097179" d="176640" r="0"/>  <S t="6003273819" d="180480" r="11"/>  </SegmentTimeline>  </SegmentTemplate>  <Representation id="root\_audio67" bandwidth="182400" codecs="mp4a.40.5" audioSamplingRate="24000"/>  </AdaptationSet>  <!-- Same as previous audio -->  <AdaptationSet id="5" contentType="audio" mimeType="audio/mp4" lang="en">  <AudioChannelConfiguration schemeIdUri="urn:mpeg:mpegB:cicp:ChannelConfiguration" value="2"/>  <ContentProtection schemeIdUri="urn:mpeg:dash:mp4protection:2011" value="cenc" cenc:default\_KID="55ff04db-d75a-dfb9-ef8a-6473ae9ac9c4"/>  <ContentProtection schemeIdUri="urn:uuid:afbcb50e-bf74-3d13-be8f-13930c783962" robustness="SW\_SECURE\_CRYPTO">  <cenc:pssh>...</cenc:pssh>  </ContentProtection>  <ContentProtection schemeIdUri="urn:uuid:9a04f079-9840-4286-ab92-e65be0885f95" value="MSPR 2.0">  <cenc:pssh>...</cenc:pssh>  </ContentProtection>  <Role schemeIdUri="urn:mpeg:dash:role:2011" value="dub"/>  <SegmentTemplate initialization="$RepresentationID$/init.mp4" media="$RepresentationID$/$Time$.mp4" timescale="90000" startNumber="807170070" presentationTimeOffset="36403">  <SegmentTimeline>  <S t="6002915623" d="178560" r="0"/>  <S t="6003094183" d="181440" r="0"/>  <S t="6003275623" d="178560" r="0"/>  <S t="6003454183" d="181440" r="0"/>  <S t="6003635623" d="178560" r="0"/>  <S t="6003814183" d="181440" r="1"/>  <S t="6004177063" d="178560" r="0"/>  <S t="6004355623" d="181440" r="0"/>  <S t="6004537063" d="178560" r="0"/>  <S t="6004715623" d="181440" r="0"/>  <S t="6004897063" d="178560" r="0"/>  <S t="6005075623" d="181440" r="1"/>  </SegmentTimeline>  </SegmentTemplate>  <Representation id="root\_audio68" bandwidth="182400" codecs="ec-3" audioSamplingRate="48000"/>  </AdaptationSet>  </Period>  </MPD> |

*Add to the end of section 5.3.3.5:*

Finally, if the content author signals the ability of Adaptation Set switching for any Adaptation Sets and intends to use @qualityRanking attributes in such adaptations sets, then such attributes shall be defined in all adaptation sets that are included in @value attribute. Additionally, such attributes shall be assigned by using equivalent ranking method applied to all representations in the included adaptation sets, and such equivalence should be signaled by including Quality Equivalence Descriptor (clause 5.8.5.13) listing the same group of Adaptation Sets in its @value attribute.

*Add after 5.10.4.6:*

**5.10.4.7 DASH Period Event**

**5.10.4.7.1 General**

DASH Period events are instructions in the content that a Period may be added by a processor, for example by an MPD Proxy or in the DASH client, by simple means. These Period events are identified by the URN "urn:mpeg:dash:event:period:2020".

A content author may use such event information to insert a Period at this presentation time following the rules in the following sub-clauses.

**5.10.4.7.2 Semantics**

|  |  |
| --- | --- |
| **Key** | **Description** |
| schemeIdURI | Set tourn:mpeg:dash:event:period:2020 |
| start\_time | Provides the media presentation time of the Period. |
| value | Provides the conditioning of the splice point  1: no splice conditioning  2: core CMAF profile constraints  3: extended CMAF profile constraints  4: reserved |
| duration | Shall be set to 0 |
| message | Provides a Period element as defined in clause 5.3.2 that includes a subset of elements and attributes. Permitted elements and attributes are documented in clause 5.10.4.7.3. |

The above values are mapped to presentation time, duration and the message of the event and may be carried in the MPD or inband.

**5.10.4.7.3 Permitted Period elements and attributes**

The following Period elements and attributes are permitted in the message, together with the processing model.

* @id: replaces the existing Period@id
* @start: Provides the @start value of the Period. If present, it shall be semantically identical to the value of start\_time. If not identical, the processor adding the Period may ignore this attribute.
* **BaseURL**: may add one or several new BaseURLs.
* **AssetIdentifier**: replaces the existing Asset Identifier
* **ServiceDescription**: adds an additional Service Description

**5.10.4.7.4 Period Insertion Processing Model**

When processing such an Event, the equivalent MPD with the added Period is generated as follows:

* If an Event is received with a new **Event@**id, then the client acts as follows:
  + The processor uses the start time of the event to determine when the Period boundary will happen. If it decides to add a Period, it does as follows:
    - The **Period**@start is set such that it matches the presentation time of the event.
    - The **Period**@id is added such that it matches the message.
  + All information on Period level is copied from the containing Period except for the information contained in the message that is processed according to 5.10.4.7.3.
  + @presentationTimeOffset is set for all Representations of this Period to the media value at Period start.

else the event is ignored.

*Change 5.10.1 to:*

Events may be provided in the MPD or within a Representation in order to signal aperiodic information to the DASH Client or to an application. Events are timed, i.e. each event starts at a specific media presentation time and may have a duration. Events include DASH specific signalling or application-specific events. DASH events are identified by scheme identifiers defined in this document. For application specific events, a scheme identifier identifies the application such that the DASH Client can forward the event to the proper application.

Events of the same type are clustered in Event Streams by the same scheme/value pair. This enables a DASH Client or the application to subscribe to an Event Stream of interest and ignore Event Streams that are of no relevance or interest.

The event is called active during the time interval starting from the event’s start time until (the event’s start time + duration). If the DASH client receives an event during the event’s active time, and if the client has not processed an equivalent event before, it is expected to immediately process the event. Annex A.13 describes the client event processing.

Two ways of signalling events are provided, namely:

— events signalled in the MPD as defined in subclause 5.10.2,

— events signalled inband in the Representation as defined in subclause 5.10.3.

Generally, the Event Stream timing model follows the timing model of a media Representation in a Period.

DASH-specific events are defined in subclause 5.10.4.