CHANGE REQUEST

DASH-IF IOP	CR	0005	rev	-	Current version:	3.4

Status:		Draft		Internal Review	X	Community Review		Agreed

Title:	Robust and Consistent Live Services
Source:	DASH-IF Live Task Force
Supporting Companies:	MLB, Qualcomm Incorporated
Category:	**C** ***Date:*** 2016-11-16

Use <u>one</u> of the following categories:
C *(correction)*
A *(addition of feature)*
B *(editorial modification)*

Reason for change:	Several issues have been identified with consistent live services. Especially the packager actions in case of errors and program changes needs some detailed overviews
Summary of change:	Update robust operation to add some more clarity Provide detailed deployment guidelines for a packager
Consequences if not approved:	Not applicable.

Sections affected:	4.8.1 – 4.8.8, 4.10 (new)
Other comments:	

Disclaimer:	This document is not yet final. It is provided for public review until the deadline mentioned below. If you have comments on the document, please submit comments by one of the following means: - at the github repository https://github.com/Dash-IndustryForum/IOP/issues (public at https://gitreports.com/issue/haudiobe/DASH-IF-IOP) - dashif+iop@groupspaces.com with a subject tag [LIVE], or Please add a detailed description of the problem and the comment. Based on the received comments a final document will be published latest by the expected publication date below, integrated in a new version of DASH-IF IOP if the following additional criteria are fulfilled: - All comments from community review are addressed - The relevant aspects for the Conformance Software are provided - Verified IOP test vectors are provided
Commenting Deadline:	December 9th, 2016
Expected Publication:	December 31st, 2016

4.8 Robust Operation

4.8.1 Background

In order to support some of the advanced use cases documented in section 2, robust service offerings and clients are relevant. This document lists the relevant ones. In clause 4.10, more detailed requirements and recommendations are provided for a specific live content generation scenario.

4.8.2 Tools for Robust Operations

4.8.2.1 General Robustness

General Guidelines in ISO/IEC 23009-1 **Error! Reference source not found.** DASH spec in A.7:

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

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

- If the DASH access client receives an HTTP client or server error (i.e. messages with 4xx or 5xx error code), the client should respond appropriately (e.g. as indicated in in RFC 7231 **Error! Reference source not found.**) to the error code. In particular, clients should handle redirections (such as 301 and 307) as these may be used as part of normal operation.

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

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

- Similarly, if the DASH access client receives an HTTP client error (i.e. messages with 4xx error code) for the request of a Media Segment, the requested Media Segment may not be available anymore or may not be available yet. In both these case the client should check if the precision of the time synchronization to a globally accurate time standard is sufficiently accurate. If the clock is believed accurate, or the error re-occurs after any correction, the client should check for an update of the MPD.

- Upon receiving server errors (i.e. messages with 5xx error code), the client should check for an update of the MPD. If multiple BaseURL elements are available, the client may also check for alternative instances of the same content that are hosted on a different server.

4.8.3 Synchronization Loss of Segmenter

In order to address synchronization loss issues at the segmenter, the following options from the DASH standard should be considered with preference according to the order below:

1. The server is required to always offer a conforming media stream. In case the input stream or encoder is lost, the content author may always add dummy content that results in a conforming bitstream. However, this may result that the DASH client is not aware of such an outage and does therefore not trigger any robustness actions.

2. Usage of the Segment timeline: In general, with every generated segment on the server, the DASH content generator writes the Segment Timeline following the rules in 4.5.2.2. Only if changes are done beyond the ones excluded, an MPD validity expiration is added. Specifically, when an encoder fails for one or more specific Representations to generate the next Segment, then the DASH content generator may add a segment in the Segment timeline.

3. Early Terminated Periods or early terminated Representations (signalled by `@presentationDuration`) as included of ISO/IEC 23009-1. Early Terminated Periods may be added that contain both **`Period`**`@start` and **`Period`**`@duration`. For Early Terminated Periods, the value of the **`Period`**`@duration` is the presentation duration in Media Presentation time of the media content represented by the Representations in this Period. The MPD is updated using the `@minimumUpdatePeriod`, i.e. the timeline is progressing. The `@presentationDuration` may be added to individual Representations to signal early termination of Representations in the Period. This permits server to signal that there is an outage of media generation, but that the service is continuing. It is then up to the client to take appropriate actions.

4. Gap filling segments: If the decoder or the playback system supports the ability to overcome losses by providing codec independent gap filling segments (such as a null segment), such an approach may be used as well. No standardized way to support this functionality exists as of today, but it is expected that MPEG will define such functionalities in the near future.

4.8.4 Encoder Clock Drift

In order to support robust offering even under encoder drift circumstances, the segmenter should avoid being synced to the encoder clock. In order to improve robustness, in the case of an MPD-based offering Periods should be added in a period continuous manner. In the case of MPD and segment-based control, the producer reference box should be added to media streams in order for the media pipeline to be aware of such drifts. In this case the client should parse the segment to obtain this information.

4.8.5 Segment Unavailability

To address signaling of segment unavailability between the client and server and to indicate the reason for this, it is recommended to use regular 404s. In addition, unless a UTC Timing has been defined prior in the MPD, the Date-Header specifying the time of the server should be used. In this case, the DASH client, when receiving a 404, knows that if its time is matching the Date Header, then the loss is due to a segment loss.

4.8.6 Swapping across Redundant Tools

To enable swapping across redundant tools doing hot and warm swaps, the following should be considered

1. the content author is offering the service redundant to the client (for example using multiple Base URLs) and the client determines the availability of one or the other. This may be possible under certain circumstances

2. Periods may be inserted at a swap instance in order to provide the new information after swap. If possible, the offering may be continuous, but the offering may also be non-continuous from a media time perspective.

3. A completely new MPD is sent that removes all information that was available before any only maintains some time continuity. However, this tool is not fully supported yet in any DASH standard and not even considered.

There is a clear preference for the bullets above in their order 1, 2 and 3 as the service continuity is expected to be smoother with higher up in the bullet list. At the same time, it may be the case that the failure and outages are severe and only the third option may be used.

4.8.7 Service Provider Requirements and Guidelines

The requirements and guidelines in subsections 8.2 to 8.6 shall be followed.

4.8.8 Client Requirements and Guidelines

The client shall implement proper methods to deal with service offerings provided in section 8.2 to 8.6.

==== CHANGE Add new section 4.10 ===

4.10 Deployment Scenarios

4.10.1 Introduction

This section addresses specifically considered deployment scenarios and provides proposed service configurations based on the technologies introduced in section 4.

4.10.2 Reliable and Consistent-Delay Live Service

4.10.2.1 Scenario Overview

A service provider wants to run a live DASH service according to the below Figure 1. As an example, a generic encoder for a 24/7 linear program or a scheduled live event provides aproduction encoded stream. Such streams typically includ inband events to signal program changes, ad insertion opportunities and other program changes. An example for such signalling are SCTE-35 [XXX] messages. The stream is then provided to one or more Adaptive Bitrate (ABR) encoders, which transcodes the incoming stream into multiple bitrates and also conditions the stream for segmentation and program changes. These multiple encoders may be used for increased ABR stream density and/are then distributed downstream for redundancy purposes. The resultant streams are received by theDASH generation engines that include: MPD generator, packager and segmenter.Typically the following functions are applied by the MPD packager:

- Segmentation based on in-band information in the streams produced by the ABR encoders
- Encapsulation into ISO BMFF container to generate DASH segments
- Dynamic MPD generation with proper customization options downstream
- Event handling of messages
- Any other other DASH related adaptation

Downtream, the segments may be hosted on a single origin server, or in one or multiple CDNs. The MPD may even be further customized downstream, for example to address specific receivers. Customization may include the removal of certain Adaptation Sets that are not suitable for the capabilities of downstream clients. Specific content may be spliced based on regional services, targeted ad insertion, media blackouts or other information. Events carried from the main encoder may be interpreted and removed by the MPD packager, or they may be carried through for downstream usage. Events may also added as MPD events to the MPD.

In different stages of the encoding and distribution, errors may occur (as indicated by lightning symbols in the diagram), that for itself need to be handled by the MPD Generator and packager, the DASH client, or both of them. The key issue for this section is the ability for the DASH Media Presentation Generator as shown in to generate services that can handle the incoming streams and provide offerings such that DASH clients following DASH-IF IOPs can support.

Hence this section primarily serves to provide guidelines for implementation on MPD Generators and Packagers.

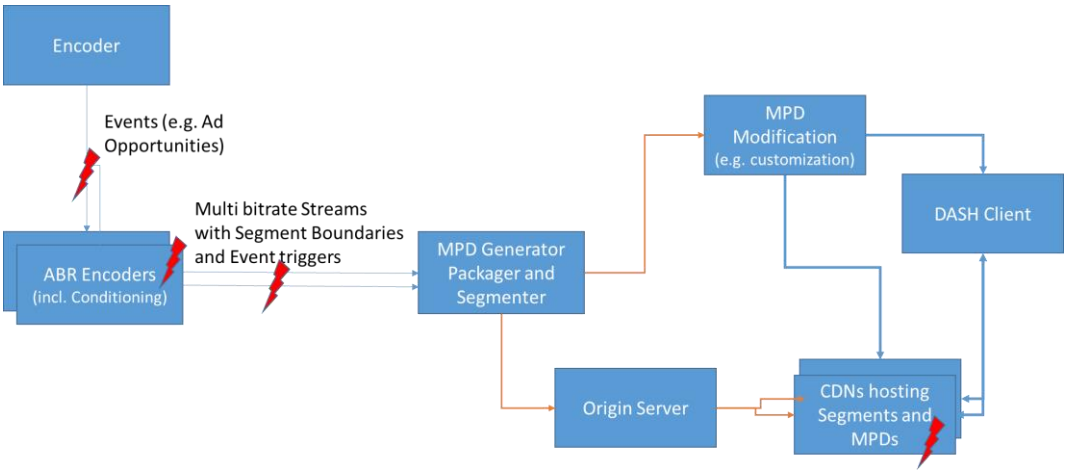

Figure 1 Example Deployment Architecture

4.10.2.2 Service Considerations

The following scenarios are considered in the service setup:

1. The distribution latency should be consistent, typically what is observed for broadcast TV services. This means that the MPD Generator should add minimum delay, and the service should be setup such that the delay between MPD generator and DASH client playout is consistent, and preferably small.

2. Program events may occur for different reasons, for example Program changes, switches from Programs to Advertisements or vice versa, media blackouts or other program changes. Such changes are typically anticipated only on short notice, i.e. within a few seconds. In the following we refer to the time that changes are announced as change lead time. The service should also provide a minimum change lead time, i.e. the smallest time in media time between the change being announced in the stream and the time between the change actually occurs. Changes may for example include one or more of the following:

    a. Number of source audio languages or formats can change. For example:

        i. Programming with English and Spanish to other content with only English

        ii. Descriptive audio may disappear / reappear

        iii. Programming with 5.1 E-AC-3 and AAC Stereo content to other content with only Stereo AAC

    b. Resolution or format of source video content can change, e.g. HD to/from SD, HDR to/from SDR, etc.

    c. Codecs may change, or at least the profile or level of the codecs

    d. The number of Representations in an Adaptation Set may change

    e. A distribution network may be changed, added or removed.

    As an example, at broadcast origination points if MPEG-2 TS is used, then the Program Map Table (PMT) typically indicates changes such changes. Typically these changes also result in discontinuities for in the media timeline.

3. The segmentation is determined by the ABR encoders. This encoding may result in occasional slight variations in segment durations during a period (as compared to the last segment in a period) due to encoding optimizations around scene changes in close proximity to the segment duration point (for example: making a segment slightly shorter or longer to align segment IDR to a scene change).

4. Unanticipated losses and operational failures or outages, possibly happen just for a single encoding (typically at the input of the encoder, but also possibly also downstream packaging). Examples are

    a. An encoder for one or more Representations or the output of an encoder fails for some time and does not produce content.

b. An encoder or the input to the encoder or the output of the encoder fails for a media component/Adaptation Set for some time and do not produce content, e.g. referring to issues as documented in Annex B.4.

c. All encoding or the input to the encoder fails for some time e.g. referring to issues as documented in Annex B.4.

In all cases an MPD can still be written and the MPD is up and running.

Also in the distribution, single Segments may be lost for different reasons and the client typically gets 404.

5. MPD updates should be minimized, whereby MPD updates includes the following aspects for every MPD request

a. Client sending uplink requests for MPDs

b. Sending full MPD with every request from the server to the client

c. Parsing and processing of MPD at the client

d. Writing a new MPD on the server if the MPD is changed

All factors are relevant to some extent, but primarily the issues a and b should be minimized.

4.10.2.3 Relevant DASH-IF IOP Technologies

4.10.2.3.1 Introduction

This document includes technologies that permit to solve the problems addressed above. We review the available technologies and justify the selection of the technology for the considered scenario. A proposed service configuration is provided in clause 4.10.3.3.

4.10.2.3.2 Consistent Latency

The scenario as introduced in 4.10.2.1 and 4.10.2.2 does not ask for very low latency, but for consistent latency. In DASH-IF IOP, latency can primarily be controlled by the following means:

- segment duration: the segment duration typically directly impacts the end-to-end latency. Smaller segment sizes provide improved latency and segments of 1-2 seconds may be chosen, if latency is an important aspect. However, too small segments may result in issues, as compression efficiency decreases due to more frequent closed GOPs in the elementary stream. In addition, the number of files/requests to be handled is higher, and finally, with shorter segments, TCP throughput may be such that not the full available capacity on the link can be exploited. Clause B.4.8 and 4.3.3.2.2 provide some guidelines on this.

- If files are available in chunks on the origin, for example due to specific encoding or delivery matters, chunked delivery may be supported. If this feature is offered, then the `@availabilityTimeOffset` attribute may be provided to announce how much earlier than the nominal segment availability the segment can be accessed.

- In order to provide tight synchronization between client and server, and therefore providing the receiver the ability to request the segment at the actual segment availability time, the availability time synchronization as defined in clause 4.7 should be provided and signalled in the MPD. Typically support for `http-xsdate` is sufficient for consistent latency support. Accurate NTP synchronization is recommended, but not required for the MPD packager or the DASH client as long as the time synchronization API is provided.

- It is proposed that a client consistently implements and joins at a segment that is slightly offset (e.g. 4 segments earlier) from the live edge segment. The exact number depends on the distribution system (for example in a fully managed environment, the offset may be smaller in contrast to best effort networks). The MPD author may support consistency by providing a suggested presentation delay in the service offering. For details on joining at the live edge, please refer to 4.3.4.4.2.

4.10.2.2.3 Unanticipated New Periods

To avoid that the clients take future segment existence for granted even if a sudden change on the service offering is necessary, the MPD service provider must set to the **MPD**`@minimumUpdatePeriod` to a low value. All Segments with availability start time less than the sum of the request time and the value of the **MPD**`@minimumUpdatePeriod` will eventually get available at the advertised position at their computed segment availability start time.

In the most conservative case, the MPD author sets the **MPD**`@minimumUpdatePeriod` to 0. Then only Segments with availability start time less than the request time are available, i.e. no promise for future segments is provided. The DASH client is forced to revalidate the MPD prior to any new Segment request. For this purpose, basically two options exists:

- Option 1) Client revalidates MPD with every Segment request according to clause 4.4.4, preferably using a conditional GET in order to avoid unnecessary downlink traffic and processing in the client.

- Option 2) Client relies on MPD validity expiration events in event messages, if content provider announces those in the MPD and by this, it can revalidate.

Note that the two methods are not mutually exclusive. More details are discussed further below.

In case of option 1 using MPD level validation, with every generated segment on the server, the DASH content generator checks the validity of the MPD offering. If still valid, no changes to the MPD are done. Only if changes are done that are no longer valid, a new MPD is written.

4.10.2.2.4 Segment Duration Variations

Variable segment durations impact the accuracy of the MPD times of the Segments. MPD times are used for the computation of the segment availability time. With variable segment durations, the segment availability times vary and can impact the DASH-IF IOPs basically provide to options to deal with variable segment durations

- Option 1)

    - Signalling of constant segment duration using `@duration`, permitting a variation of +/- 50% of the segment duration. According to 3.2.7.1, for each media segment in each Representation the MPD start time of the segment should approximately be *EPT* - *PTO*. Specifically, the MPD start time shall be in the range of *EPT* - *PTO* - 0.5**DUR* and *EPT* - *PTO* + 0.5**DUR* according to the requirement stated above.

      Note that the encoder should provide segments of a virtual segmentation that adheres to this rule. However, there may be reasons that the encoder does break this rule occasionally.

    - If the DASH packager receives a segment stream such that the drift can no longer be compensated, then a new Period should be added, that adjusts the parameters for the segment availability computation, but also signals that the Period is continuous as defined in 4.3.3.3.2. Note that this is done for all Representations in the MPD and a change of the MPD is happening, i.e. this needs to be anncouned. However, no segment parsing is necessary for the client.

- Option 2) Following the rules in 4.5.2.2 and using the Segment Timeline to accurately signal the different segment durations. If the segment duration changes, then the `@r` attribute of the last **S** element in the Segment timeline is terminated and a new S element is added to the MPD with the new segment duration. Note that this results in a change of the MPD. The client should determine such changes independent of MPD updates by detailed segment parsing to obtain the earliest presentation time of the segment and the segment duration.

4.10.2.2.5 Losses and Operational Failures

One of the most complex aspects are occasional operational issues, such as losses, outages, failovers of input streams, encoders, packagers and distribution links. Section 4.8 provides detailed overview on available tools that should be used by network service offering and clients in order to deal with operational issues. Several types of losses may occur as shown in Figure 2:

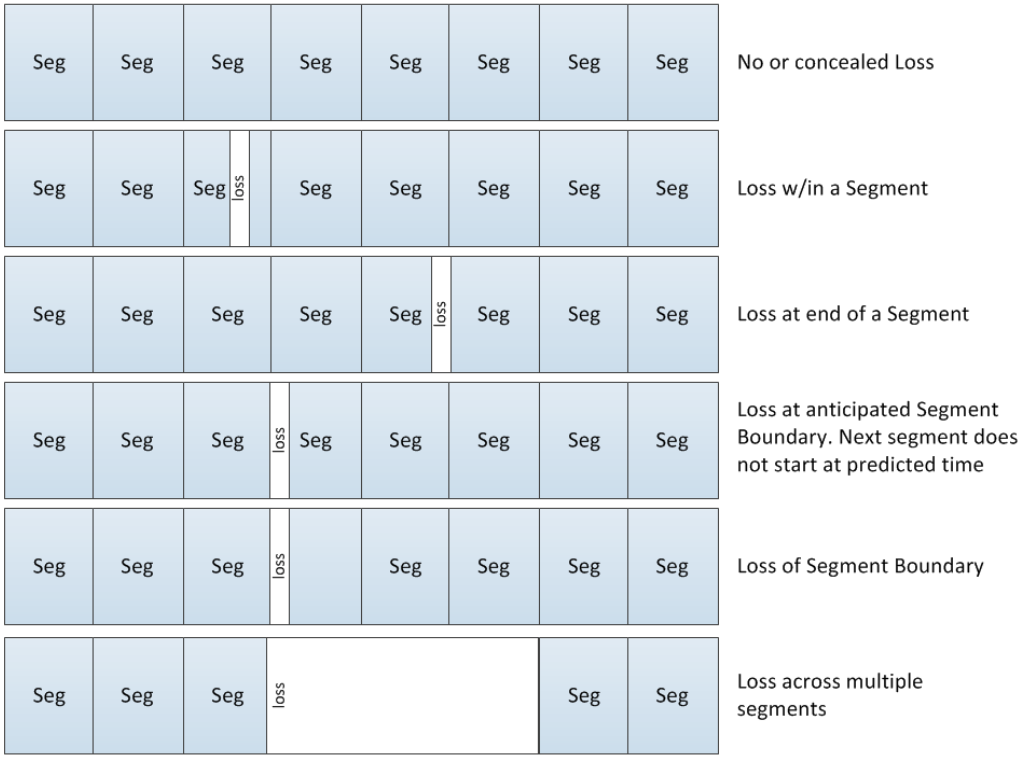

Figure 2 Loss scenarios

Losses may occur in the middle of a Segment, at the end of a Segment, at the start of a new Segment. At the elementarty stream level, losses may be within a compressed access unit (AU), producing a syntactically corrupt bitstream, or may be the result of the ABR encoder simply not encoding a source frame in which case the duration of the prior AU is extended producing a conforming bitstreaming. Losses may impact an entire Segment or may just impact a part of the Segment. Typically, service oriented losses will occur until the next Random access point, i.e. a loss is to be signaled from the start of the lost sample up to the next random access point, typically conindicing with the start of a new Segment.

In order to deal with this, the MPD packager basically has the following options that are not mutually exclusive:

- Option 1: Operation with Main live profile expecting main profile clients. In this case the Segment Timeline is used to signal gaps in the media. For details see below.
- Option 2: Operation with simple live profile expecting simple profile clients. In this case the early terminated Period is used to signal gaps in the media. For details see below.
- Option 3: Gap filling segments. Note that Gap filling segments are not part of the IOP recommendations as the DASH client would be unaware that such gaps are actually happening. The DASH client can not take proper measures, such as switching to a Representation that is not under loss and does provides the actual content. Therefore, gap filling segments should be used by the content author carefully taking this into account.

  Note: At the time of writing we are aware that MPEG is addressing the issue of gap filling segments. It is expected that at foreseeable time an update on this issue will be provided.

In addition to the above, the content provider may offer to provide the same content on different Base URLs. In this case, the temporary non-availability may be signaled as well through the MPD.

4.10.2.2.6 Minimizing MPD Updates

MPD updates, the frequency of MPD updates and the actions included in MPD updates are different ones, and their effects may have different impacts on deployments. To avoid confusion on the generally overloaded term, some more details are discussed in the following section. In non-DASH adaptive streaming solutions, MPD updates result in the following additional processing and delivery overhead:

1. The client sends an uplink requests for the MPD. At least from a CDN perspective, this is issue is considered less critical, typically the bounds of operation are reached by throughout, not by the number of requests.

2. The server needs to send a full MPD with every request, which for itself causes overhead from all the way of the origin server to the client. This is in particular relevant if the manifest contains a list of URLs, and some timeshift buffer is maintained.

3. Yet another aspect is the regular parsing and processing of the manifest in the client. Whereas the processing is likely less of a burden, the consisteny across two parsing instances is relevant and requires to keep state.

4. MPD updates may also result in writing a new MPD on the server. This may be less problematic for certain cases, espescially for unicast, but it results in significant overhead if DASH formats are used for broadcast.

DASH-IF IOP provides different means to avoid one or the more of the above issues. Assuming that the **MPD**`@minimumUpdatePeriod` is set to a low value for reasons documented above, then issues mentioned above can be addressed by the following means in DASH-IF IOP

1. Client Requests: can be avoided by signalling inband that an MPD is has expired. The most obvious tool is the use of Inband Events with MPD expiry. However, this requires inband events being added during packaging.

2. Sending Full MPD: Instead of requesting the full MPD, the client can support this operation by issuing a conditional GET. If the MPD has not changed, no MPD needs to be sent and the downlink rate is small. However, this requires the usage of `@duration` or SegmentTimeline with `@r=-1`.

3. MPD Parsing and Processing: This can be avoided by using either of the solutions documented above.

4. MPD writing on server: This goes hand-in-hand with 2, i.e. the usage of `@duration` or **`SegmentTimeline`** with `@r=-1`.

Generally, DASH-IF IOP provide several tools to address different aspects of minimizing MPD updates. Based on the deployment scenario, the appropriate tools should be used. However, it is preferable that DASH clients support different tools in order to provide choices for the service offering.

4.10.2.3 Proposed Service Configuration and Generation of the MPD and Segments based on a "Segment Stream"

4.10.2.3.1 Introduction and Assumptions

The core concept is the availability of a segment stream at the input to a packager. The segment stream may be made available as individual segments or as boundary markers in a continuous stream. In addition, the stream may contain information that is relevant for the packager, such as program changes. The segment stream determines for each segment the earliest presentation time, the presentation duration, as well as boundaries in program offerings.

Furthermore, it is assumed that multiple bitrates may exist that are switchable. In the following we focus on one segment stream, but assume that in the general case multiple bitrates are available and the encoding and segment streams are generated such that they can be switched.

The high-level assumptions for the service are summarized in 4.10.2.1. Based on these assumptions, a more detailed model is provided.

- A segment stream is provided for each Representation. The segmentation is the same for Representations that are included in one Adaptation Set. Each segment *i* has assigned a duration *d*[*i*] and an earliest presentation time *ept*[*i*]. In addition, the segment stream has a nominal segment duration *d0* that the ABR encoders attempts to maintain. However, variation may occur for different reasons, documented above.

- Losses may occur in the segment stream, spanning a part of a segment, multiple segments, a full segment and so on. The loss may be in one Representation or in multiple Representations at the same time (see above for more discussions).

- The latency of the time that the segment is made available to the DASH packager and that it is offered as an available segment in the MPD should be small, i.e. the segment availability time should be shortly after the time when the full segment is received in the DASH packager. Any permitted delay by the MPD Packager can be view as additive to change lead time and may therefore improve efficiency and robustness, but may at the same time increase the end-to-end latency.

- Changes in the program setup may occur, that signal changes as discussed in 4.10.2.2. A change is possibly announced with a time referred to as change lead time. Note that signal changes such as SCTE-35 only indicate where a change may occur, it does not indicate what type of change will occur.

The different scenarios are summarized in Figure xxx.

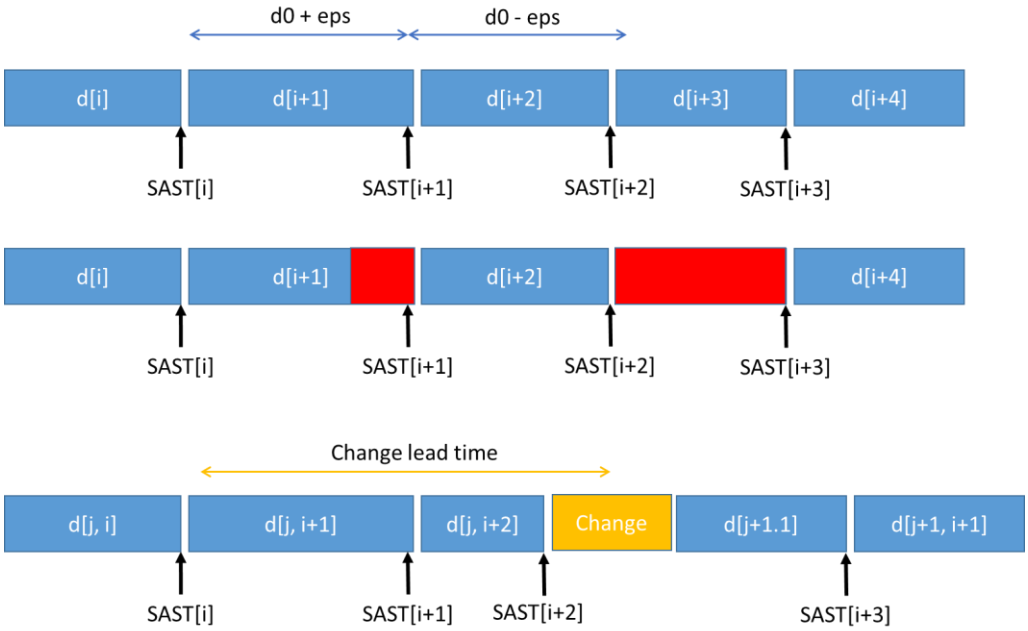

Figure XXX: Different properties of a segmen stream.

NOTE: We consider adding some explicit examples such as SCTE-35 messages or CMAF related information. Input is welcome.

Based on the discussons in 4.10.2.1 and 4.10.2.2, proposed service configuration for such a service are proposed. The service configuration differentiates two deployment scenarios:

1) Clients implementing the simple live client, i.e. no emsg support and no segment parsing is implemented. This is documented in clause 4.10.2.3.2.

2) Clients implementing the main client, i.e. emsg is supported and segment parsing is implemented. This is documented in clause 4.10.2.3.3.

In the following, reference is made to technologies in section 4.10.2.2.

4.10.2.3.2 Service Configuration for Simple Live

Assuming that the input stream is a segment stream with the properties documented above is received by the DASH packager.

The DASH packager may operate as follows:

- The `@minimumUpdatePeriod` is set to a value that is equal or smaller than the change lead time provided by the segment stream.

- The `@timescale` of the Adaptation Set is set to the timescale of the included media

- The `@duration` attribute is set such that the nominal duration d0 is documented in the MPD for this Adaptation Set.

- `$Number$` is used of segment templating.

- With incoming segments of the segment stream, a new segment is generated by the DASH packager and the DASH packager checks the validity of the MPD offering. If still valid, no changes to MPD are done. Only if changes are done that are no longer valid, a new MPD is written. Specifically,
    - The MPD start time of the next segment must be in the range of *EPT - PTO* - 0.5**DUR* and *EPT - PTO* + 0.5**DUR* with *DUR* the value of `@duration`.
    - If this is not fulfilled a new Period is written that includes the following:
        - The **`Period`**`@start` is set such that the MPD start time is correct.
        - The `@presentationTimeOffset` is set to the EPT of the first segment
        - The `@startNumber` is set to the first segment in the new Period.
        - The Adaptation Sets are continued by providing Period continuity signallng with each Adaptation Set.
- when an encoder fails for one or more specific Representations to generate the next segment, then the DASH content generator
    - terminates the Segment with the last sample in the segment, (which is possibly corrupted)
    - generates a new MPD as follows:
        - The `@minimumUpdatePeriod` is set to `0`.
        - If all or at least many Representations fail, the Period`@duration` is set to the value of the media time in the Period that is still available.
        - If only a subset of the Representations fail, the `@presentationDuration` for the last segment is set to the value of the last presentation time in the Representation that is still available.
        - By doing so, the content provider basically informs the DASH client that for the duration of the Segment as announced, no media is available. The DASH client revalidates this after every Segment duration. The MPD is not changed on the server until either the decoder resumes or the Media Presentation is terminated.
        - If the `@minimumUpdatePeriod` is long, then the client may request non-existent segments, which itself may then trigger that the DASH client revalidates the MPD. If the DASH client has the possibility, it should add the '`lmsg`' brand as a compatibility brand to the last generated segment. In addition, when the segment is distributed over HTTP, the HTTP header should signal the content type of the segment including the compatibility brand 'lmsg'. If the DASH client can identify this, it is expected to refetch the MDP and may by this means observe the early terminated Period or Representations.
    - Only after the encoder resumes, a new MPD is written as follows:
        - A new Period is provided with **`Period`**`@start` according to the value of the new Period. The `@presentationTimeoffset` of the Representation of the Period shall match the the earliest presentation time of the newly generated Segment. If appropriate, Period connectivity should be signaled.
        - The `@minimumUpdatePeriod` is set again to the minimum change lead time.
- when a program change is announced, generates a new MPD as follows:
    - The `@minimumUpdatePeriod` is set to `0`.
- When the program change occurs
    - Write a new MPD with all the parameters

- Reset the `@minimumUpdatePeriod` is set to a value that is equal or smaller than the change lead time provided

4.10.2.3.2 Service Configuration for Main Live

Assuming that the input stream is a segment stream with the properties documented above is received by the DASH packager.

The DASH packager may operate as follows:

- The `@minimumUpdatePeriod` is set to 0.

- The `@timescale` of the Adaptation Set is set to the timescale of the included media

- The segment timeline is used. Addressing may used: `$Number$` or `$Time$`.

- The MPD is assigned an **`MPD`**`@publishTime`

- With incoming segments of the segment stream, following the rules in 4.5.2.2 the DASH Packager uses the Segment Timeline to accurately signal the different segment durations. If the segment duration changes, then the `@r` attribute of the last **`S`** element in the Segment timeline is terminated and a new S element is added to the MPD with the new segment duration. The values `@t` and `@d` need to be set correctly:

    - `@r` of the last segment element may be set to -1. In this case a new MPD is only written if the segment duration changes

    - `@r` of the last segment element may be set to the actual published number of segments. In this case a new MPD is written for each new segment

- Whenever a new MPD is written, the **`MPD`**`@publishTime` is updated.

- when an encoder fails for one or more specific Representations to generate the next segment, then the DASH packager

    - terminates the Segment with the last sample in the segment (may be corrupt)

    - adds `emsg` to this last generated segment. The MPD validity expiration is set to the duration of the current segment or smaller. This `emsg` may be added to all Representation that have observed this failure, to all Representations in the Adaptation Set or to all Representations in the MPD. The content author should be aware that if the emsg is not signaled with all Representations, then there exist cases that a switch *to* the erroneous Representation causes a request to a non-existing Segment. That loss would be signaled in the MPD, but the client is not aware that an update of the MPD is necessary.

    - The `emsg` should be added to all Representations that announce that they carry the message as an inband stream.

        - NOTE: Do you consider that it should be mandated to add the emsg to all Representations. The benefit is that even corner cases will not result in a request for a lost segment. The downside is that all encoders need to be coordinated and that for all clients that stick with a non-corrupt Representation will do an unnecessary MPD update.

    - The MPD is updated on the server such that the last generated segment is documented in the Segment timeline and no new **`S`** element is added to the timeline.

    - Only after the Representation(s) under loss resumes, a new **`S`** element is written with **`S`**`@t` matching the earliest presentation time of the newly generated Segment. The DASH client with it next update will resume and possibly take into account again this Representation.

    - If the encoder does not resume for a specific Representation over a longer time, it is recommended to terminate this Period and remove this Representation at least temporarily until the encoder resumes again. Period continuity should be signaled.

- when the program change occurs

- adds `emsg` to this last generated segment. The MPD validity expiration is set to the duration of the current segment or smaller. This `emsg` shall be added to all Representations that announce the Inband Event stream for the MPD validity expiration.
- Write a new MPD with all the parameters

- Whenever a new MPD is written, the **`MPD`**`@publishTime` is updated.

4.10.2.4 Client Support Considerations

4.10.2.4.1 Introduction

Generally the client should support the rules in this section for the specific clients.

4.10.2.4.2 Client Requirements for Simple Live

The client shall follow the details in clause 4.3.4 and 4.4.4. In addition, the DASH client is expected to handle any losses signalled through early termined Periods.

4.10.2.4.3 Client Requirements for Main Live

The client shall follow the details in clause 4.3.4 and 4.5.3. In addition, the DASH client is expected to handle any losses signalled through gaps in the segment timeline.

The DASH client having received an MPD that signals gaps is expected to either look for alternative Representations that are not affected by the loss, or if not possible, do some appropriate error concealment. The DASH client also should go back regularly to check for MPD updates whether the Representation gets available again.