 ISO/IEC JTC 1/SC 29/WG 03 N0087

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

**Document type:** Output Document

**Title:** Insert Document

**Status:** Approved

**Date of document:** 2020-10-16

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

# Expected action: None

# Action due date: None

**No. of pages:** 2 (with cover page)

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

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

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 03 MPEG SYSTEMS**

**ISO/IEC JTC 1/SC 29/WG 03 N** **0087**

**October 2020, Virtual**

|  |  |
| --- | --- |
| **Title** | **Exploration on Encoder and Packager Synchronization** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **19833** |

# Introduction

# Segmented Ingest Solutions

## DASH-IF Live Ingest Protocol

### Architecture

The architecture as envision for practical usage with redundant components. The goal is that the client gets a consistent manifest, regardless if requesting from origin A or origin B.

Ingest 1 Ingest 2 DASH/HLS

Encoder A

Encoder B

Packager A

Packager B

Origin A

Origin B

Client N

### DASH-IF Live Ingest Protocol

The dash-if live ingest protocol [dash-if live media ingest](https://dashif-documents.azurewebsites.net/Ingest/master/DASH-IF-Ingest.html) can be used to do redundant & synchronized encoder ingest either using:

Interface 1: different encoders push CMAF tracks

Interface 2: encoder push DASH or HLS and possibly CMAF fragments/

We consider interface 1, as by using only cmaf segments and tracks, and generating the manifest later, not errors in the manifest need to be corrected in case of upstream errors.

### Solution

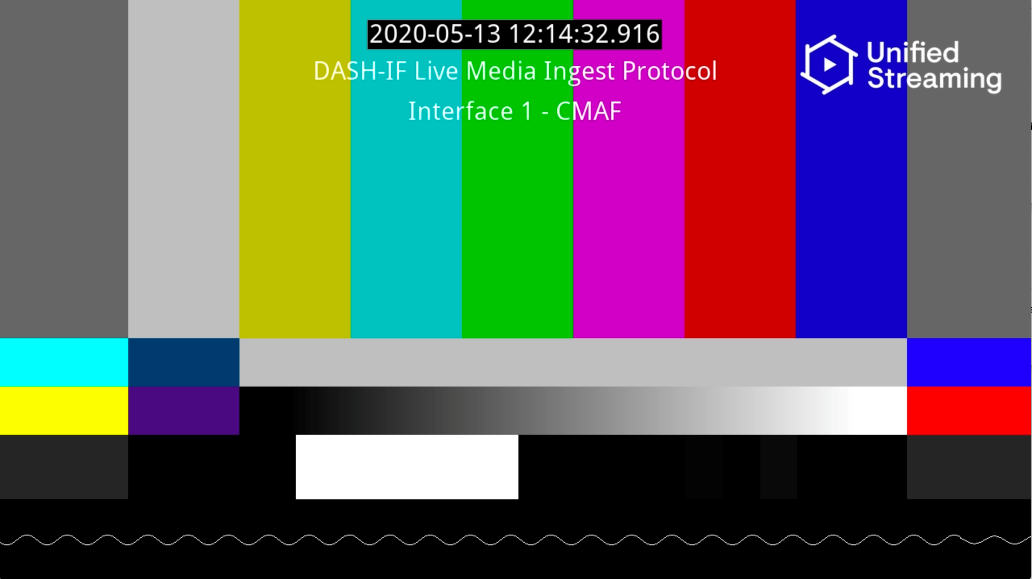
We use the [dash-if live media ingest](https://dashif-documents.azurewebsites.net/Ingest/master/DASH-IF-Ingest.html) CMAF ingest and combine origin and packager servers (just-in-time packager). The encoders use a fixed/mutually known anchor time **(possibly unix epoch or a pts in the input MPEG-2 TS that is exchanged between encoders and is configurable)** and a **fixed segment duration**. To allow redundant encoders to join, leave and join again an encoder pushes CMAF segments **anchor + K \*segment duration** with baseMediaDecodeTime of **anchor + K \* segment\_duration.** By using the same anchor in the input signal, this could achieve encoder synchronization given that:

1. Encoder A and encoder B share the same TS or SDI source with timing information
2. Encoder A and Encoder B produce the same time dependent content (e.g. SMPTE error bar with clock)
3. Encoder A and Encoder B are clock synchronized within possible 100 ms accuracy which can be achieved using common synchronization

The receiver can detect duplicate segments by identical tfdt or if the time is embedded in the segment URL from the segmentTemplate + segmentTimeline S(t,d). The packager generates an MPD, such as using the CMAF profile for dash, using the shared anchor as the period start, and the segment timeline will be updated when segments are received. This way a consistent manifest can be constructed at both origin A and origin B. By using the timing from the original source no race condition should occur between origin A and Origin B.

### Demo

The demo is available at [encoder sync github](https://github.com/unifiedstreaming/live-demo-cmaf/tree/encoder_sync) and is a docker compose based demo using public function repository, the demo implements the architecture with two encoders (ffmpeg), a single origin/packager (Apache/Unified Origin). The demo should look something like this when it is ran, the test image should detail the encoder pushing the live stream aswell (either encoder B or A). Right now single sample fragments are used, and the origin can repackage the stream, but in future the demo will use multiple sample fragments aswell.



### Conclusion

Using DASH-IF live media ingest technical specification encoders can be synchronized:

* Using synchronized source (e.g. MPEG-2 TS with pts)
* Not using overly strict clock synchronization withing 100 ms should be fine
* Fixed segment durations (splices can be sync samples inside a segment, packager creates segment boundaries)
* Fixed anchor (e.g. based on the time source or another epoch)
* CMAF instead of DASH, generating a consistent manifest at the received using segmentTimeline based indexing, for number based indexing the number K could be used (segment\_time - time-achor/fixed\_segment\_duration)

The benefit as compared to using DASH is that race conditions or different mpds are avoided due to upstream errors. We recommend the encoder synch AhG to consider this approach and consider possible extensions to target different requirements.

### Specific Recommendation based on Demo for Encoder synch

1. Use shared time anchors (possibly epoch), and use that for media presentation timeline , period start etc.
2. Use fixed segment durations (packager can still change this to account for splicing etc.)
3. Use DASH-IF ingest interface1[1] based on CMAF [2]

### References

[1] DASH-IF Live media ingest specification (DASH-IF technical specification): <https://dashif-documents.azurewebsites.net/Ingest/master/DASH-IF-Ingest.html>

[2] ISO/IEC Common Media Application format ISO/IEC 23000-19:2020

Information technology — Multimedia application format (MPEG-A) — Part 19: Common media application format (CMAF) for segmented media

## Proposed Segmented Ingest protocol

### Introduction

Traditionally, the interface between linear transcoders and packagers has been MPEG-2 TS multicast over UDP. This approach works fine in provisioned networks and was adapted for virtually segmented output using the CableLabs EBP and SCTE ATS (SCTE 214-4). However, this approach is becoming more and more problematic.

In modern ecosystems MPEG-2 TS is relegated to legacy distribution mechanisms where typically MPEG-2 (ATSC 1.0) and AVC (QAM/IPTV) are the only feasible codecs. Dependency on a MPEG-2 TS transcoder-to-packager hop for an ISO-BMFF-based service makes it much harder to deploy new technology. For example, IMSC1 subtitle support in ISO-BMFF is fairly widespread, while support for DVB-TTML (IMSC1 in MPEG-2 TS) is hard to come by.

The simplest alternative is having a transcoder output ISO-BMFF directly. This is a commonplace solution which makes the physical transcoder server also an origin. The problematic aspect of such a solution is that any issue resulting in a flood of HTTP GET requests to the transcoder/origin combo may take the complete channel offline for millions of viewers.

Once a more complex redundant deployment is considered, the encoder/packager combo becomes more complex, as all MPD creation logic, as well as alternative content, ad insertion, inter-packager synchronization, et. al. now need to be handled within the same software package provided by the transcoder vendor. 

Figure 1: Synchronized packager ecosystem

A simple solution to the first problem is pushing segments, similar to the logic used in the traditional MPEG-2 TS multicast. This is as simple as issuing HTTP POST requests from the transcoder to a packager. The complexity problem can be addressed by separation of concerns – the transcoder outputs segments and the essential metadata, while the packager becomes a “smart origin” which accepts segments, generates manifests, and serves them to the CDN.

This protocol was proposed by Unified Streaming and evolved into the DASH-IF Live Media Ingest Protocol (<https://dashif-documents.azurewebsites.net/Ingest/master/DASH-IF-Ingest.html>). The protocol defines two interfaces, segment-only CMAF-based Interface 1, and Interface 2, which outputs DASH or HLS.

The proposal below attempts to create a simple segmented protocol based on the above DASH-IF Live Media Ingest Protocol, and addressing some additional use cases

### Proposal

#### Protocol

##### Initialization

Initialization is performed “at the beginning of time” and is done for the whole transcode job.

The client issues HTTP POST of an MPD describing the set of representations the client intends to provide.

This MPD shall contain a single Period and conform with the following restrictions:

1. Only $Time$-based addressing can be used,
2. Only relative URLs are allowed in BaseURL and SegmentTemplate elements
3. MPD should obey CMAF profile (ISO/IEC 23009-1:202X 5th edition)
4. (additional restrictions tbd)

NOTE: the MPD posted by the client is intended for the packager (“smart origin”), and is not intended for consumption by a DASH client. The packager may also merge multiple MPDs if parts of the presentation are encoded using different encoder instances.

#### Message exchange

The following exchange is for each representation. It happens only after the server returned a 2xx response to the POST of the MPD.

1. Client posts initialization segment (CMAF header) to the path identical to the initialization segment URL appearing in the corresponding Representation element.
2. Client posts media segments, with one HTTP POST per segment. If HTTP/1.1 is used, chunked transfer mode is recommended. The path part of for the URL shall match its relative URL derived from the Representation element.

NOTE 1: the above can be achieved using the “timeline extension” technique, where the $Time$ value is inferred from `tfdt` and not specified explicitly in the SegmentTimeline element of the MPD.

NOTE 2: persistent HTTP connection is possible (and will be the default) in the case of POST-per-segment. A combination of POST-per-representation and HTTP chunked transfer encoding approach is (a) not atomic, (b) cannot be upgraded to HTTP/2 or higher, and (c) is more problematic when failover / load balancing is needed.

1. In case encoder input is lost, the client shall output a segment containing all valid frames it has, and signal an upcoming discontinuity using the 0x01 flag in the `mfid` box proposed below. In case of chunked transfer mode, the `mfid` box (described below) with the discontinuity indicator will appear in the last chunk of the segment.
2. If there was no input for the duration exceeding the value of the MPD@maximumSegmentDuration attribute, the client will keep outputting Missing Content segments (`miss`, as defined in 4th ed. of MPEG DASH). The segments will have the duration of either the maximum segment duration or till the first frame of the newly regained input.

NOTE: If the encoder generates valid output (e.g. frozen frame), this output will be included in the `miss` segment.

1. If the input is lost for a specific representation, and it is expected to return, then the client will post `miss` segments with it will send `miss` segments with additional brand `nntr` (“nothing new to report”).
2. If the communication is terminated by the client, the last segment shall have a `lmsg` brand (“last segment”, as defined in MPEG DASH). If there is no input, the segment will be the Missing Content Segment.

#### ISO-BMFF restrictions

1. For video, VPS/SPS/PPS should be carried in `mdat`. For example, `avc3` and `hev1` can be used for AVC and HEVC, as opposed to `avc1` and `hvc1`
2. `prft` box shall be present and should correspond to the studio time whenever possible. The value there should either correspond to the acquisition time of the picture (studio time) or the item at which the frame entered the encoder.

##### Handling SCTE 35

1. SCTE 35 and ID3v2 shall be passed within an `emsg` box appearing just prior to any `moof` box within the segment.

##### Common Encryption

1. There may be workflows where in-transit security offered by HTTPS is insufficient and a proper content protection mechanism is required. In the above case, the initialization segment shall contain the `pssh` box.

##### Redundant operation

1. If a received segment indicates forward discontinuity, the server may request the corresponding segment from a redundant server. It will issue an HTTP GET using $Time$ addressing based on `tfdt` value of this segment. [NOTE: this implicitly assumes that the segment boundaries of the incoming segments are synchronized. MPD request will be needed otherwise]
2. The server will keep requesting segments from the redundant server until it receives a segment with backwards discontinuity. [If synchronization is assumed, the segment following the backwards-discontinuous segment will be aligned with the redundant server]

### Movie fragment identifier box

#### Definition

Box Type: 'mfid'  
Container: MovieFragmentBox or MovieBox  
Mandatory: Yes  
Quantity: Exactly one

#### Syntax

aligned(8) class MovieFragmentIdentifierBox  
 extends FullBox('mfid', flags, 0){

unsigned int(64) segment\_number

unsigned int(32) chunk\_number

string mpd\_id

string period\_id

string switching\_set\_id

string representation\_id

}

#### Semantics

flags

0x01: this fragment ended due to temporary loss of media, rather than by authoring intent. Next segment number may not necessarily be an increment of the previous segment number

0x02: this fragment starts after a temporary loss of media, no continuity from the previous segment is expected.

0x004 -- moov

segment\_number: segment number corresponding to a fragment. The number may start at an arbitrary value, the number zero shall only be used in initialization segment (i.e., only when this box is within the `moov` box).

chunk\_number: CMAF chunk number corresponding to a fragment. The first fragment of a segment shall always have chunk number of zero.

The following shall hold for two consecutive fragments i and i+1:

* If there is no discontinuity indicated by the 0x01 flag in fragment i or 0x02 in fragment i+1, then the following holds:

if (chunk\_number[i+1] == chunk\_number[i] + 1 ) {

segment\_number[i+1] = segment\_number[i]

} else {

segment\_number[i+1] = segment\_number[i] + 1

}

mpd\_id : Value of the MPD@id attribute in the corresponding MPD. The value shall be identical across all representations belonging to the same channel

period\_id: Value of a Period@id attribute.

selection\_set\_id: Value of the corresponding AdaptationSet@id attribute

representation\_id: Value of the corresponding Representation@id attribute

### Guidelines for use of the `mfid` box

The `mfid`box shall appear at the beginning of each media segment and in an initialization segment. When a segment contains more than one fragment (e.g., in case of CMAF chunks) it is possible to add it to the `moof` box starting the chunk. This may be needed e.g. when chunks are delivered using HTTP/2 server push.

The `mfid` box is intended for the packager rather than the end client. Its purpose is segment identification and grouping. There is nothing a DASH or HLS client can do with the information contained in this box, hence a packager may remove it.

### Conformance

#### MPD construction

Multiple partial MPDs may be posted by encoders generating different representations belonging to the asset. They will all have the same value of MPD@id.

A full MPD shall be constructed as follows:

#### MPD reconstruction

Given an MPD M with MPD@id = X and a collection of segments, the following process creates an MPD M’ such that M’ is functionally equivalent to M.

1. Iterate over all segments and group segments with MPD@id=X, and identical period, selection set, and representation ID. Order the collection by segment id. Each collection then corresponds to a representation
2. Iterate over representations with MPD@id=X, and identical period and selection set ID. Each collection corresponds to an adaptation set
3. Group adaptation sets where MPD@id=X, and identical period values
4. Create MPD element with MPD@id = X and availabilityStartTime set to the start of the epoch
5. Create period elements, deriving Period@start value from the `prft` box of the first segment of a period.
6. Create adaptation set elements, populating them with information derived form the initialization and media segments. Use SegmentTimeline and $Time$-based addressing
7. Create representation elements , populating them with information derived from the initialization and media segments.

# Encoder Synchronization

## Introduction

In case of high-value wide-distribution channels and events there is a need for a very high degree of reliability. When reliability is addressed from the adaptive streaming point of view, the link between the origin and the streaming client gets most of the attention. However, this link is not the only source of a failure, and loss of a mezzanine feed or a transcoder has catastrophic consequences as it affects all viewers of the channel.

Geographically distributed encoding facilities is a common mitigation technique. As a US-centric example, a mezzanine source from Tokyo can be sent to a data center location in San Francisco (West Coast) and another one to a data center in New York (East Coast). This makes the service resilient to loss of a transcoder or of a complete data center (e.g. due to a power outage).

Additional layer of reliability can be achieved by using different mezzanine sources – e.g. one (primary) data center is using high-quality high-bitrate lossless mezzanine source over dedicated fiber, while the other is receiving a compressed lower-quality mezzanine source via satellite or the public Internet.

The ultimate goal of such designs is to sustain loss of a single transcoder or its network connectivity, a loss or a temporary glitch in a mezzanine source, or even a loss of complete data center without a service disruption. An additional goal may also load balancing where the client may elect to download a segment produced by either of the redundant transcoders.

In order to allow glitch-less switchovers between redundant encoder outputs we need identical outputs. There is no need for the outputs to be bit-identical – what is needed is complete time alignment. This means that segments output by each transcoder have to start at the same frame and have identical earliest presentation time. This, in turn, also requires identical IDR placement.

The encoders may output into multiple origins, and even be accessed through different CDNs. We assume that this is implemented in DASH using multiple BaseURL elements at the same level.

Another use case is multi-encoder setup. This can happen when a single server chassis is capable of encoding a subset of representations, but not the complete ladder.

Lastly, pure encoder synchronization is insufficient – it is important to have functionally identical MPDs in addition to synchronized segments.



Figure 2 Synchronized encoder ecosystem

## Requirements

1. For any combination of segments coming from transcoders T0..Tn, the playback shall be continuous and independent of the MPD location;
2. There may be more than two redundant transcoders to synchronize.
3. The sources may be delayed relative to each other (e.g. direct fiber vs transcoded satellite feed). Note that this may result in different NTP times in `prft` boxes unless these are adjusted as well.
4. The sources may be differently transcoded. For example, the primary sources may be compressed with a lossless codec such as JPEG2000 or JPEG-XS while the secondary source can be a high-rate HEVC video
5. There is no guarantee that any time signaling is present in the mezzanine files

## Possible approaches: encoders

### Source with timing

In many cases the source feed can carry timing. In case of MPEG-2 TS it can be carried using means such as SCTE 35 time\_signal() or CableLabs EBP. In case of HEVC or AVC mezzanine streams timing can be carried in an SEI message. There is a multitude of ways absolute time can be carried together with MPEG-2 TS.

Having established a precise correspondence between absolute time and frame, it is possible to coordinate an IDR location. Coordinated IDR locations imply segment alignment.

One approach for synchronizing IDR locations would be creation of a rule where in-stream time is directly translated into an IDR location. For example, the first frame after an even second would always be an IDR. This approach is very simple and resilient; however, it does not let the encoder use variable-duration segments and may create complications when ad breaks do not fall on segment boundary.

A different approach would be to explicitly coordinate between transcoders. For example, transcoders can exchange messages with time of the IDR frames, with one encoder being the leader.

### Identical MPEG-2 TS input

If the input is the same MPEG-2 TS stream, passing through PTS values can establish the same correspondence as absolute time. Then the same approach of either creating a PTS-to-boundary correspondence or coordination using messaging would apply

### Synchronized input

If (a) the same input enters the encoders at the same time (up to a few millisecond difference), and (b) encoders have clocks synchronized to the same time source, absolute frame acquisition time can be similarly used for coordinating segment boundaries. The millisecond difference here accounts for at the least arrival time at the NIC.

### Picture-based synchronization

In absence of either timing in input streams or precise clock and input synchronization, boundary synchronization can be achieved by examining the frames. If the frames are bitwise-identical, cryptographic hashes such as MD5 can be used to identify frames. If the frames are different (e.g. one source is high-rate MPEG-2 video off the satellite, the other is JPEG2000 over direct fiber), then features such as color histograms and histograms of edges can be used for synchronization.