Text

Description automatically generatedISO/IEC JTC 1/SC 29/WG 03 N0231

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

**Document type:** Output Document

**Title:**  Exploration on encoder and packager synchronization

**Status:** Approved

**Date of document:** 2021-05-25

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

**No. of pages:** 12 (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 N0231**

**May 2021, Virtual**

|  |  |
| --- | --- |
| **Title** | **Exploration on encoder and packager synchronization** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **20293** |

# Introduction and problem description

## 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.

Using 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 layers 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 be 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 support for encoder synchronization and guidelines for redundant operation. A current way to achieve this is by mandating 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. The encoders may output into multiple origins, and even be accessed through different CDNs. One way how this is supported in MPEG-DASH is by using multiple BaseURL’s in the Media Presentation Description. 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, this packager synchronization, is also in scope of this exploration. This document summarizes the status of the ongoing exploration on these problems.



Figure 2 Synchronized encoder ecosystem with CMAF as an example output format of the encoder and DASH from the origin (HLS or smooth streaming are other possible client protocols, dash is just an example), the orange line indicates some of the scope of the Encoder synchronization standardization for inter encoder communication. The intermediate formats for encoder output are also in scope.

## Problem description

In practice, multiple ingest sources (like encoders or encoder/packager combos) and receiving entities (like packagers or origin servers) are often used for redundancy purposes. This requires that multiple ingest sources and receiving entities work together in a redundant workflow to avoid interruptions when some of the components fail. An example workflow is show below, where there are three encoders and two packagers providing redundancy and preferably seamless recover from a failover event.

Note that encoder/packager synchronization should be codec and potentially format agnostic. That is, whatever methods are developed and adopted by the standard should work for any modern codec (AVC, HEVC, VVC, etc.) and format (DASH, CMAF, HLS).

Shape

Description automatically generated with medium confidence

# Requirements and use cases

## Initial requirements (to be completed, solicit contributions)

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).
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 signalling is present in the mezzanine files
6. Encoders and origins may fail and restart later again
7. Solutions should not use proprietary boxes in the ISO Base Media file format, as was common for example in Microsoft smooth streaming

Editor’s note: This is initial list of explicit requirements stated in input documents, the list is to be completed, input & discussion is needed.

## Use cases for transcoder and encoder synchronization

Several use cases are in scope of this exploration, below a categorization is made based on input synchronization type to the distributed encoders.

### Source input of Transcoder or Encoder carries 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, can be useful for achieving synchronization and alignment of key frames.

### Identical (MPEG-2 TS) input

If the input is the same (e.g. 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-segment boundary correspondence or coordination using messaging may apply

### Synchronized input

If (a) the same input enters the encoders at the same time (up to a few milliseconds 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.

### Absence of timing information in the input

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 colour histograms and histograms of edges can be used for synchronization.

Editor’s note: This is initial list of use cases, input is welcome, prioritization and selection of use cases may be needed.

## Use cases for packager and origin synchronization

### Multiple transcoders pushing a full presentation to one or more origins

In this use case, more than one encoder is pushing full presentations to one or more origin servers. Origins may be in different domains and not aware of each other. Origins transmit manifests that are inter-changeable. Origins may not be clock synchronized withing 50 milliseconds accuracy. Origins may transmit content using different protocols such as DASH, HLS or others. Origins may fail and rejoin later. Encoders may fail and rejoin later.

### Multiple transcoders pushing partial presentations to one or more origins

In this use case, more than one encoder is pushing full presentations to one or more origin servers. Origins may be in different domains and not aware of each other. Origins produce manifests that are inter-changeable. Origins may not be clock synchronized withing 50 milliseconds accuracy. Origins may transmit content using different protocols such as DASH, HLS or others. Origins may fail and rejoin later. Encoders may fail and rejoin later.

### Redundant operation and failover

Similar as above, both encoders of receivers (origins) may fail and restart again later, join and/or leave a session.

Editor’s note: This was not discussed with detail in the meeting, so this topic needs some discussion in the next meeting, input is welcome.

Editor’s note: Requirements related to the use cases need to be made explicit.

# Protocols, formats and architectures

This section details protocols and architectures to be considered.

Editor’s note: With more discussion these can become requirements and use cases.

## Reference architecture

A reference architecture is assumed where:

1. One or more encoders/transcoders are receiving input (encoder input format)
2. One or more origins/packagers are receiving streams form the one or more encoders in the encoder output format
3. The output of the origin/packagers is used, to deliver the content, normally, via a content delivery network to the clients

A more detailed reference was brought up during the meeting, it is defined in DASH-IF IOP for ad insertion: <https://dashif.org/docs/CR-Ad-Insertion-r8.pdf>. It shows very well the upstream blocks in streaming architecture.

Editor’s note: More details on encoder input format and upstream blocks are needed as these are outside the DASH-IF IOP scope, discussion with SVA, SCTE, ATSC, DVB and SMPTE is recommended to complete this.

Editor’s note: A more detailed architecture for encoder synchronization may need to be developed, perhaps with the starting point from <https://dashif.org/docs/CR-Ad-Insertion-r8.pdf>.

## Encoder/transcoder input protocols and formats

The following encoder input formats have been considered so far, MPEG-2 TS, JPEG 2000 JPEX XS, (HD-) SDI, secure reliable transport, Zixi.

Editor’s note: This list needs to be completed based on input from organizations mentioned in previous section and industry participants.

## Encoder/transcoder output protocols and formats

Several encoder/transcoder output formats and timing descriptors that may be supported include the following.

### DASH-IF live ingest protocol

Two closely related protocol interfaces are defined: CMAF Ingest (Interface-1) based on fragmented MP4 and DASH/HLS Ingest (Interface-2) based on DASH and HLS. Both interfaces use the HTTP POST (or PUT) method to transmit media objects from an ingest source to a receiving entity. These interfaces support carriage of audio-visual media, timed metadata and timed text. Examples of workflows using these interfaces are provided. In addition, guidelines for synchronization of multiple ingest sources, redundancy and failover are presented. the original spec in HTML format and is available [here](https://dashif-documents.azurewebsites.net/Ingest/pull/143/DASH-IF-Ingest.html). DASH-IF welcomes issues to be reported in the [issue tracker](https://github.com/Dash-Industry-Forum/Ingest/issues). This update 1.1 is under community review and expected publication is august 2021. Clause 6.9 provides text

Editor’s note: Protocol update to version 1.1. is still under review and comments are still welcome, several implementations such as an FFmpeg based one are available.

### Segmented ingest protocol

Another segmented ingest protocol was proposed in Oct. 2020, most of the proposed additions have been merged with the updated DASH-IF specification that can be found [**here**](https://dashif-documents.azurewebsites.net/Ingest/pull/143/DASH-IF-Ingest.html)**,** however some additions to the file format may still be up for discussion in MPEG. More details are provided in Section 5.

### Time descriptors

In addition to a format based on ISO-BMFF movie fragments, there have been proposals to include additional timing descriptors, such as defined in SCTE-35, Apple, QuickTime that may be carried in ISO-BMFF structures. Examples brought up during the meeting include UTC time provision in QuickTime, SMPTE timecodes (in the mp4) for precise video time identification. Media files may also need to be self-describing, i.e., side information/files usage is not always feasible/practical.

Editor’s note: This list needs to be completed based on input from organizations mentioned in previous section and industry participants, also it should be discussed what timing descriptors are popular and why.

## Origin output protocols and formats

These include at least MPEG-DASH (ISO/IEC 23000-1), HLS as defined in RFC 8216, but other protocols are not precluded.

Editor’s note: in the ideal case, encoder synchronization support should not have any impact on clients, except improving the quality of the received presentation, i.e. the intention is to make no normative changes that affect clients

## CMAF storage format

Another slightly different, but very much related topic in scope of this exploration is the storage of CMAF contents (including the grouping of the tracks). In many cases this also poses requirements on the format/protocol in 3.3. A CMAF storage format may apply to cases of storing live archives and large VoD collections. Requirements on CMAF usage may be slightly different as compared when targeting client devices directly, and synchronization information may also need to be maintained in the archive/storage format. Discussions on this have introduced adding identifiers in CMAF tracks and using the DASH core profile for CMAF that can accurately describe a CMAF presentation.

Editor’s note: Due to the close relationship, storing CMAF presentations and live archives should also be considered in the exploration, more input on this is welcome.

Editor’s note: This should be discussed and corresponding requirements should be derived.

# Organizational aspects

During the meeting several points regarding organization of the work were brought up. This section tries to summarize some of the key points.

The DASH-IF developed the media ingest protocol, that already has some support for encoder source and origin/packager synchronization, MPEG has a liaison with the DASH-IF and can requests updates to be made, as was done after the MPEG meeting in October, which led to an update in the DASH-IF ingest specification currently under review (see section 3). The update under review contains several improvements for encoder synchronization. DASH-IF, however, is an industry forum and not a standards body, hence formally standardizing some aspects of DASH-IF ingest in MPEG/ISO can be considered as part of this activity in MPEG.

MPEG is stronger on core formats, such as file format, CMAF, thus specific additions and profiling may be in scope of the MPEG work on encoder synchronization. The outcome may result in extensions to MPEG-A, ISO-BMFF, MPEG-DASH or other specifications. It was mentioned specifically that it is not the goal to develop an extensive set of APIs but leave that to the implementors/vendor specific solutions. However, recommended use of the ISO Base Media File Format and MPEG-DASH are in scope.

As both DASH and HLS are in scope coordination with organizations working on HLS is also needed such as CTA-WAVE.

RIST and SRT are other potential input formats for encoders, while any changes on these protocols are out of scope, some of their characteristics may be considered, and informing the organizations developing these protocols on this work may also be part of the effort.

To further and clarify the scope of the standardization, a workshop will be organized to develop reference architecture and answer some of the questions and open issues in this document. Members from all these organization with an interest will be invited.

# Technical evidence of solutions and demonstrations

Several solutions have already been proposed to MPEG. These solutions highlight both technically feasible solutions as evidence of solving this problem for general cases, and existing work on this problem. These solutions may serve as an anchor/reference solution, and the standardization should address encoder synchronization beyond the scope of these solutions addressing all key requirements brought forward during the discussion/exploration that are accepted as critical.

The next subsections detail the proposed solutions.

## Encoder synchronization with CMAF ingest and epoch locking

This contribution was brought forward in contribution m56779.

### DASH-IF live ingest protocol

The DASH-IF live ingest protocol dash-if live media ingest can be used to do redundant and synchronized encoder ingest either using:

A) Interface-1: Different encoders push CMAF tracks

B) Interface-2: Different encoders push DASH or HLS and possibly CMAF fragments

Note that the updated version under community review contains an explicit clause:

DASH-IF Live Media Ingest Protocol (dashif-documents.azurewebsites.net) in 6.9 on encoder synchronization. Review and comments are welcome. In m56779, we use option A) to push segments with aligned timestamps and segment boundaries.

### Solution

We use DASH-IF ingest Interface-1 and combine origin and packager servers (just-in-time packager). The encoders use a fixed/mutually known anchor time (possibly UNIX epoch) and a fixed segment duration. To allow redundant encoders to join, leave and join again an encoder pushes CMAF segments with baseMediaDecodeTime of anchor + K \*segment duration with baseMediaDecodeTime of anchor + K \* segment\_duration as close as possible to the current time. This results in segment synchronization given that:

a) Encoder A and encoder B share the same TS or SDI source with timing information, such that the of frame time relative to the anchor can be done consistently.

b) Encoder A and Encoder B produce the same time dependent content (e.g., SMPTE error bar with clock), or the frame presentation time maps consistently with the per frame input timestamps.

c) Encoder A and Encoder B are clock synchronized within a possible 100 ms accuracy which can be achieved using common synchronization, as to consistently compute K based on the current time.

d) By using constant segment durations, frame boundaries are aligned allowing identical timing and alignment between audio and video segments.

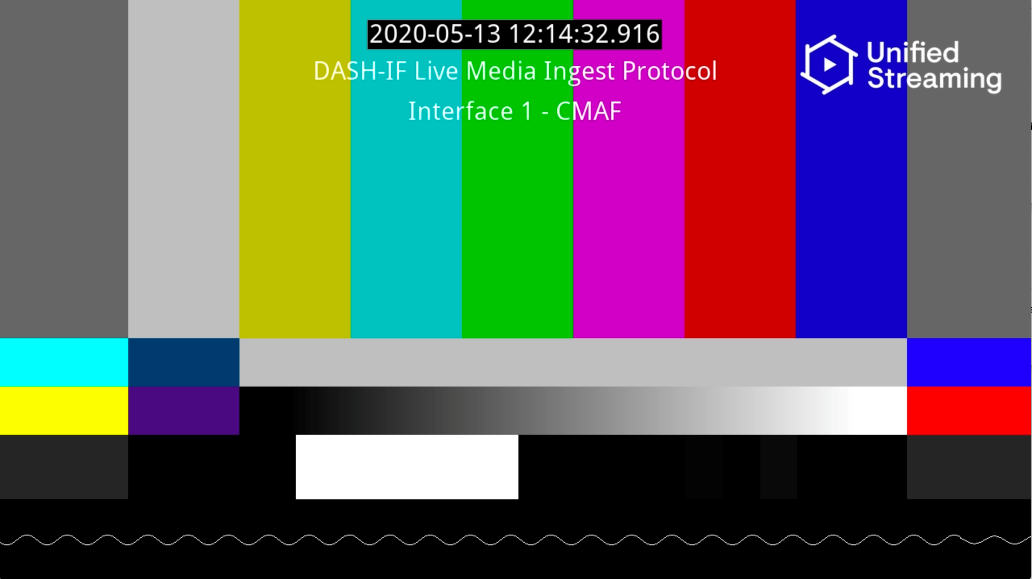
The receiver can detect duplicate segments by identical baseMediaDecodeTime or if the time is embedded in the segment URL from the segmentTemplate + segmentTimeline S(t,d). The packager generates an MPD, such as using the CMAF profile for DASH using the shared anchor as the period start, and the segment timeline will be updated when the 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.

An example script for computing the offset for audio and video to be relative to UNIX epoch taking different timescale of audio and video into account can be done with the following script, it is available on [Unified CMAF Encoder Sync Logic - Replit](https://replit.com/@JamieFletcher2/Unified-CMAF-Encoder-Sync-Logic)

The advantage of this approach is that there is no need to use multiple BaseURL’s.

### 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 run. The test image should detail the encoder pushing the live stream as well (either encoder B or A). Right now, single sample fragments are used, and the origin can repackage the stream, but in the future the demo will use multiple sample fragments as well. Note: A license key may be required to run this demo independently!



### Other Benefits and Application Areas

Using CMAF and constant segment duration and fixed anchor can also be beneficial for example to synchronize tracks coming from other third parties or processes. Examples include CMAF timed metadata tracks and MPEG-B part 18 timed metadata tracks. Other examples include timed text, or parts of the bitrate ladder or different codecs. These can be produced by separate threads or processes that can be synchronized in a similar manner allowing for distributed and synchronized deployment. This corresponds to the use case where different encoders are only producing a part of the presentation (e.g., encoders may only encode part of the ladder, the subtitles and or timed metadata)

### C**onclusion and summary**

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

* Using sources carrying timing information
* Synchronized source generation using SMPTE error bars and clock
* 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)
* Script to compute offsets from Unix Epoch for both audio and video timescales

The benefit as compared to using DASH ingest (direct) is that race conditions or different MPDs are avoided, e.g., by generating the manifest later upstream errors need not propagate to clients. The specific method is referred to as EPOCH locking. It is especially shown that CMAF is a strong backend format to achieve stream synchronization.

Some requirements raised:

1. Use shared time anchors (possibly epoch), and use that for media presentation timeline in ISOBMFF, period start, etc.
2. Use fixed segment durations (packager can still change this to account for splicing etc.).
3. Use DASH-IF ingest Interface-1 based on CMAF.
4. Use case of CMAF for dual encoder sync. may be documented in MPEG, it should even work if the DASH-IF ingest is not followed strictly.

### Some additional comments on this approach

The approach can also be used with DASH-IF live media ingest interface-2 if it also implements the format requirements on timing and segment boundary alignment. However, in this case, the MPD may not be able to account for upstream errors corrected by the dual ingest downstream, and manifest may still need to be corrected/updated.

## Proposed solution: stream synchronization protocol

This solution was contributed in m5692.

### Architecture

Multiple synchronized transcoders processing the content form a *constellation*. In the simplest case, a constellation comprises two encoders. In more interesting cases there may be multiple encoders encoding different representations (using DASH terminology) of the same content.All encoders within the constellation are subscribed to some pub-sub system (e.g., a message queue system such as ActiveMQ or a plain UDP multicast).

Each encoder can be in an unsynchronized mode (e.g., it has input but is not yet synchronized with the rest of the constellation), in a synchronized mode, or in a divergent mode (encoder has no input or its input differs from some of the constellation members).

An encoder in synchronized mode sends periodic messages (defined below) describing the frame and its timing.

An encoder starts in an unsynchronized mode, subscribes, and listens to the messages from the synchronized encoders in the constellation until it reaches synchronization. Synchronized encoders accept [at least] IDR location and decoding timestamp (`tfdt`) from the leader encoder. For each encoder, the leader encoder is defined as encoder with highest ID number and matching input.

### Message format

#### SSP message

|  |  |  |
| --- | --- | --- |
| Syntax | No. Of bits | Mnemonic |
| ssp\_message() {  sender\_id  constellation\_id  num\_frames  num\_descriptors  length  reserved = 0  for (i = 0; i < num\_frames; i++ ) {  decode\_timestamp  type  reserved = 0  for (j = 0; j < num\_descriptors; i++ ) {  sps\_descriptor()  }  }  } | 128  128  8  8  8  52  64  3  7 | uimsbf  uimsbf  uimsbf  uimsbf  uimsbf  bslbf  uimsbf  bslbf  uimsbf |

#### Time descriptor

|  |  |  |
| --- | --- | --- |
| Syntax | No. Of bits | Mnemonic |
| time\_descriptor() {  descriptor\_type  length  time\_source  time\_carriage  time\_seconds  time\_nanoseconds  utc\_offset  }  } | 8  8  3  4  48  32  16 | uimsbf  uimsbf  uimsbf  uimsbf  uimsbf  uimsbf  uimsbf |

#### Hash descriptor

|  |  |  |
| --- | --- | --- |
| Syntax | No. Of bits | Mnemonic |
| hash\_descriptor() {  descriptor\_type  length  hash\_type  num\_horizontal\_splits\_plus1  num\_vertical\_splits\_plus1  for (j = 0;  j < num\_horizontal\_splits\_plus1\* num\_vertical\_splits\_plus1;  j++ ) {  **hash**  }  } | 8  8  8  4  4  N | uimsbf  uimsbf  uimsbf  uimsbf  uimsbf  uimsbf  uimsbf |

#### Histogram descriptor

|  |  |  |
| --- | --- | --- |
| Syntax | No. Of bits | Mnemonic |
| histogram\_descriptor() {  descriptor\_type  length  histogram\_type  num\_horizontal\_splits\_plus1  num\_vertical\_splits\_plus1  num\_bins  bin\_width\_log2  reserved = 0  for (i = 0; i < num\_bins; i++ ) {  for (j = 0; j < 3; j++ ) {  **count**  }  } | 8  8  8  4  4  16  4  12  bin\_width | uimsbf  uimsbf  uimsbf  uimsbf  uimsbf  uimsbf  uimsbf  uimsbf |