ISO/IEC JTC 1/SC 29/WG 03 N0364

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

**Document type:** Output Document

**Title:** Explorations on storage, archiving and content management of CMAF files

**Status:** Approved

**Date of document:** 2021-10-15

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

**No. of pages:** 10 (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 N0364**

**Oct. 2021, Virtual**

|  |  |
| --- | --- |
| **Title** | **Explorations on storage, archiving and content management of CMAF files** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **20841** |

# Introduction

CMAF has been developed as the preferred track format for streaming media. As a consequence, more and more content is becoming available in this format stored on disks, cloud storage, content delivery networks and more. CMAF is not only practical for streaming using DASH, HLS or other delivery protocols, but also shows a potential for storage and archiving purposes.

CMAF defines a hypothetical application model and constructs for streaming that map nicely to delivery protocols like DASH and HLS. However, when storing CMAF content, grouping of tracks and identification of CMAF constructs is not that straightforward. Content management systems may need to handle CMAF content as it gets distributed and cached/stored through a series of packagers, origin servers, object/block storage and edge servers. We first list some challenges one might face for using CMAF as a storage format and then provide some ideas.

The main goal is to make the community aware of these challenges, provide standard-based solutions, seek input from the practitioners and eventually agree on a set of best practices to promote interoperability.

Editor’s note: Should the binding be only for CMAF track files or also for other constructs?

In this document, we will use the following terms to specifically refer to the purpose of the manifest in that context:

* **S-Manifest (S-MPD)**: Storage manifest file that is used to describe when storing CMAF content (files). An S-MPD can be created from a collection of CMAF files additionally containing identifiers, and globally unique identifiers.
* **D-Manifest (D-MPD)**: Delivery manifest file that is used to deliver CMAF content from a server to CMAF clients.
* **(CMAF) Identifiers:** Information in the CMAF header of a CMAF track file to store CMAF grouping identifiers or hints, content properties/hints and/or track properties. Identifiers should have a schemeIdUri to signal the identifier type and a valuesignalling the identifier value. Relational structural identifiers can support the generation of the manifests through identifying the MPD structural relationship between CMAF files. Global identifiers are unique for each CMAF track and MPD stored within the content management system. This can be used to separately create CMAF content and create/attach MPDs in the content workflow.

# Approaches

## M53812 (Segment reverse look-up, MPEG 130)

The proposal is to generate and add a UUID for each CMAF track and a UUID for the associated (but yet to be created) manifest. Additionally, a set of IDs to indicate the hierarchical level within the manifest structure (Period, Group, AdaptationSet, Representation) is created and this is tied to CMAF structures such as switching set, selection set and presentation. For each CMAF track, redundant information across CMAF segments is stored in an initialization segment. If the information changes depending on the segment, e.g., Period ID timestamp, then this information could be stored in an ‘emsg’ or ‘meta’ box or other means to be determined. This gives an approach from examining the selected directory or subdirectory of CMAF tracks to create or associate a manifest and placement of segment information within that structure. If the manifest is yet to be created, then UUIDs and IDs provide a bottom-top reference to the manifest and relational information between tracks when it is created. If the manifest already exists, then it provides the ability to look up the manifest(s) from the examination of the CMAF track(s).

Editor’s note: During MPEG 130 it was mentioned that also an identifier for aligned switching sets or DASH adaptation set switching groups could be added. This would be for switching sets or adaptation sets that are expected to be switchable and conform to the CMAF definition of aligned switching set. This construct exists in both DASH (adaptation set switching) and CMAF and can be used for example when you have an AVC and HEVC switching set or switching set with different key protection.

Editor’s note: When introducing identifiers, should it also introduce a source identifier to solve that issue?

A screenshot of a cell phone

Description automatically generated

Figure 1 Relationship between the manifest and its segments.

A screenshot of a cell phone

Description automatically generated

Figure 2 Orphaned content stream or asset with information to do reverse lookup.

A simple example approach on creating a manifest from a set of CMAF tracks with UUIDs and IDs:

1. Read all the segments and create a list of unique representation UUIDs.
2. For each representation UUID, create a list of media segments with this UUID, sorted according to their timestamp (e.g., the value of the tfdt box or the first PTS).
3. Go over all the initialization segments (in case of ISO-BMFF, this would be all files containing the `moov` box). Create an MPD dictionary where a single MPD UUID is a key, and a list of [period ID, representation UUID] tuples is the value.

4. For each MPD UUID:

1. create the MPD element
2. For each period ID, create a period element
3. For each period:
4. go over all the initialization segments and group representations with the same switching set (adaptation set) ID into the same adaptation sets
5. go over all the lists of the media segments within the selection set and validate that for any i-th segment in representation X, i-th segments in any representations Y and Z in this switching set, the segment start time will be identical.

Editor’s note: This contribution presents DASH construct identifiers. It should be considered if instead CMAF identifiers should be based on CMAF constructs. An example could be as given in Table 1 (to be discussed in future meetings).

Table 1: Example CMAF identifiers and mapping to DASH identifiers.

|  |  |
| --- | --- |
| MPD ID | CMAF Presentation ID |
| Period ID | CMAF PresentationID |
| AdaptationSet  Switching Group ID | Aligned Switching Set ID |
| AdaptationSet ID | SwitchingSet id |
| Representation ID | CMAF track id (not to confuse with track\_id) |

## m54475 (MPEG 131)

Here, we deal with a specific use case of interest to the industry, that is keeping a live streaming archive using CMAF storage format. Increasingly, live streaming is moving away from MPEG2-TS to CMAF. This means that live encoders will be pushing CMAF over HTTP instead of MPEG2-TS over other defined protocols such as RTP or UDP.

In a live streaming scenario, one or more encoders are pushing live streaming content to one or more origin servers, used to stream out that content. The origins are responsible for distributing the content to clients by responding to the HTTP requests.

The archive is rather valuable, as in efficient deployments the archive can be used to implement catch-up services, VoD offerings or sub-clipping based on the archive. However, to successfully deploy and use the archive it must be structured and stored using a standardized format, possibly guidelines for storage, and mainly the synchronization between track files is important to implement the CMAF application model.

CMAF

archive

Client A

Streaming Origin

CMAF Encoder

Client B

Streaming Origin

CMAF Encoder

CMAF

archive

The problems with keeping a live archive are several. A protocol for posting CMAF content or DASH content with CMAF segments is described in the DASH-IF live media ingest specification [1] and is out of scope. It is assumed that CMAF content is received dynamically (i.e., a live scenario), and the goal is to store the content effectively such that it can be stored once and effectively monetized in different ways.

1. Redundant and synchronized encoders pushing synchronized CMAF tracks that can be used to form switching sets, aligned switching sets, **encoder/track/fragment synchronization**
2. Updating the live archive each time a new segment is received **live updating**
3. Organizing the archive (naming conventions, directory structure) **organization of the archive**
4. Methods for quickly seeking and searching/indexing through the archive

## Encoder configuration and time configuration

Use a common time base for sample decode times and base media decode time tfdt relative to epoch (1-1-1970) to allow different encoders to push synchronized content. Encoders are either synchronized producing content or use information from their input (PTS, SDI) that will be mapped to a time relative to epoch.

To enable a configuration with redundant encoders, one can set the start time of the ingest session and the default segment duration. This assumes that the ingest uses fixed segment durations (in case of splicing a sync sample, there is a smart way to fix the numbering and get back to the fixed scheme, namely to compensate the durations before and after the splice to add up to 2 \* *frag\_dur*). Any encoder joining should then start sending fragments at *start\_time* + K \* *frag\_dur* for an integer K where start\_time + K \* frag\_dur > now, and should as a consequence be in sync.

By having all tracks using the same timeline origin, tracks and sub-clips can be combined and queried/created based on the time relative to epoch, for example using a vbegin and vend, or MPD anchors as defined in DASH. In case a VoD asset is created presentationTimeOffset can be used to align the start time of the media segments with the period start.

## Updating the live archive

This is very simple, for any new segment/fragment received it is appended to the CMAF track. New CMAF headers may be used to reset or refresh the CMAF header of the track. The sender may send an S-Manifest to describe all the contents, possibly using a static MPD with a SegmentList. Alternatively, the S-Manifest is not received and the upload is annotated with CMAF identifiers or well formed (in this case based on CMAF box restrictions groupings can be made to SwitchingSets, etc.). In some cases, a database is kept summarizing the status of all received segments and tracks and/or a S-Manifest.

## CMAF storage format

The archive is kept using CMAF storage/archiving format in [1]. Each representation may be stored as a CMAF track and CMAF fragments are appended to a CMAF header. In case a representation needs to be reset, a CMAF header can be resent to the receiving origin server. As in [1] the kind box can be used to send additional information like track roles, switching set ids etc., the storage format can keep that stored. In DASH-IF ingest spec, the schemIdUri that is used for switching set grouping is:

|  |
| --- |
| *urn:dashif:ingest:switchingset\_id* may be urn:mpeg:cmaf:switchingset\_id in storage format |

For track roles the following urn is used:

|  |
| --- |
| urn:mpeg:dash:role:2011 may be urn:mpeg:dash:role:2011 |

One could use the naming scheme from DASH where bandwidth, representation id are used for the track files, it would also be good if the representation id contained information about the SwitchingSet, etc. but this is not mandatory. Alternatively, directory structures could be used. Other storage considerations from m54370 should be applied.

## Indexing in CMAF archive

One problem, described in more detail in contribution m54476, is that at some point in the live session updating a segment index segment would be needed. Putting the segment index at the end is not good as offsets in bytes can only be positive, hence, it cannot reference backwards the movie fragments like mfra.

Moving the segment index to beginning would need a shift of the live archive track file, or appending at the beginning (not supported in most file systems). This would imply a large overhead on the disk/memory.

Continuous updating of segment index would therefore not be practical at all. Creating the segment index at the end of the live session when converting VoD to static is possible, but the functionality of seeking may already be needed during the live session. Alternative segment index or segment index in separate files may be explored to that means. Currently a light database may provide a solution for this problem.

## Other issues

Other issues relate to fault tolerance in case the receiver fails. This is out of the scope of this proposal.

The problems with indexing may be added as a note, they may be discussed in file format. In addition, they may be considered for CMAF TuC/DuC as well, as CMAF does not support efficient indexing/seeking in live content due to the drawback of the segment index box. The question is whether that feature is desirable for CMAF and others are interested.

# Identified issues

## No explicit multiplexing:

CMAF track files contain a single media track to make it easier to access content independently and allow the D-MPDs to combine the tracks as desired for late binding. However, grouping and combining tracks is less explicit for storage. Multiplexed content is often preferred by the content providers.

There are multiple levels at which binding can be achieved with different resilience against loss. One can use a naming (including directory structure) convention and/or an S-MPD file. One can also use track type (?) and metadata boxes in the track files to include certain information (identifiers) about each track file.

Editor’s note: as multiplexing is both used for grouping of tracks as for joint transmission (physical multiplex), Do we need full multiplex support (sample level multiplexing) or is fragment-level or track-level multiplexing is sufficient?

## What if there is no manifest file or the manifest file is lost:

The D-MPD file is used by the client for downloading CMAF content. The stored CMAF content, however, might be an archive from a live presentation or a source content for which manifests are customly generated for specific groups. When storing CMAF, an S-MPD file is not necessarily and always available. Manifests are not CMAF specific.

Editor’s note: Could one use naming conventions of track files or identifiers in the meta box?

## CMAF media segment type identification:

CMAF defines chunks, fragments, segments as segment types. These can optionally be identified by styp boxes. However, this box is optional for broad compatibility. However, in case this box is not present, how would one identify the media segment type? Would it make sense to add alternative modes of signalling?

For storage, the styp boxes should be mandatory (if media segment identification is needed).

Editor’s note: How can this be signalled using the manifest?

## Content identification:

How could one identify that two CMAF tracks have the same underlying source content? Typically, the filename would be used for this or multiplexing (see above). Would it make sense to have something in CMAF that would allow detection of files based on the same content, i.e., a content identifier?

In a DASH MPD this can be done. The details are described in the content model. It is always very difficult to say what is the same and what is different. An obvious example is whether an HD version of a content is the same as its UHD version.

Editor’s note: Knowing the source content is identical makes it easier to apply and check the grouping structures of CMAF content, i.e., adhering to the same source content is also explicitly mentioned in the CMAF specification for definition of structures. Having identifiers will make it easier to generate the manifest later in case it is lost, but such identifiers may be additional properties to CMAF track files.

## Switching set grouping:

How could one detect that two tracks belong to the same switching set? CMAF defines switching set constraints, stating equivalent properties that need to be satisfied to be in a switching set, however, how does one know that tracks satisfying the constraints have been intended to be part of the same switching set by the author? One way we proposed in prior work is by using the kind box with a switching set identifier.

Referring to the DASH Profile for CMAF content (Amd.1), one can add all the CMAF track files in the same switching set into one Adaptation Set.

Editor’s note: How to generate the manifest based on the CMAF content when the manifest is not yet available or lost?

## Aligned switching sets:

How could one know that two groups of tracks or two tracks belong to an aligned switching set? Typically, aligned switching sets relax switching set constraints by allowing different sample entry. Example of aligned switching set could be AVC and HEVC switching sets and or switching sets with different encryption (different keys). However, from stored CMAF track files we cannot derive if they belong to an aligned switching set. Using a kind box with an identifier for aligned switching sets may again be a possible solution.

Referring to the DASH Profile for CMAF content (Amd.1), one can use segmentAlignment with the same integer number in the given AdaptationSet.

Editor’s note: How to generate the manifest based on the CMAF content when the manifest is not yet available or lost? DASH CMAF assumes that grouping of CMAF aligned switching sets but how can this be known based on the CMAF content? Additional identifiers may be needed.

## Selection sets:

CMAF defines selection sets, for this some additional identification might be useful as well.

Editor’s note: A possible solution is as follows: Selection sets should be signalled by multiple Adaptation Sets in the S-MPD file. The group attribute of AdaptationSet is used to identify selection sets when using CMAF presentation. But how do we do this when the manifest needs to be generated based on the CMAF content as described earlier?

## CMAF presentation:

How does one know a group of tracks belong to the same CMAF presentation? Typically, this would be based on file and directory structure information. Again, an identifier carried in a track file might be useful to identify a CMAF presentation.

Editor’s note: A CMAF presentation should be added to one MPD with a single period. One may add AS identifiers or program information.

## Storage and directory hierarchy:

What is the recommended storage hierarchy on disk or in the cloud for storing CMAF content? What is the recommended naming scheme/convention for files?

Editor’s note: One can use directory structures and labelling.

## Delivery of CMAF content:

What is the recommended practice for CMAF track storage to optimize delivery in different workflows? These are workflows using different protocols, pull based, push based, etc. The storage format should natively support different streaming protocols. Delivery should include scenarios of asset (CMAF tracks and manifests) from encoders, packagers, origins, CDNs, edge servers. Manifests could be created, modified, distributed and stored by separate means in these workflows aligning with applications such as nDVR.

Editor’s note: Requirements should be more explicit.

## Searching and seeking in stored CMAF content, fast and granular access:

Access of CMAF content for searching or processing is important. Segment index can help seeking through CMAF track files and should probably be recommended for storage.

Note that segment timeline in S-MPD (with $Time or $Number) cannot be used to access content in a track file independently without additional indexing or subdivision of the track file. The $Time$ or $Number$ needs to be in the file name but this filename is fixed for all segments in a CMAF track file (in this case it will be the track file name). This indexing would therefore require each segment in a separate file with a name including the $Time$ or $Number$, but then they are not CMAF track files anymore. If representation is a track, accessing a chunk or fragment in a track with segment timeline is not possible without additional indexing. For storage, access with sidx byteranges (or alike) and identification by styp in CMAF track might be sufficient for many applications.

Editor’s note: SegmentTimeline should be used in S-MPD. Segment indexes can also be used as a separate file or combined with the track files.

Editor’s note: For storing live archiving would it be possible to consider sidx at the end of the stream?

## Metadata storage:

DashEventMessageBox is one way to store inband metadata in CMAF, in addition CMAF supports timed metadata tracks. For storing CMAF content is the duplicate storage of event messages across adaptation sets sensible for most workflows? What about fast seeking and searching in metadata?

Editor’s note: Requirements should be more explicit.

## Language and bitrate:

Language and bitrate in the track file should be useful as it can be used later to regenerate D-MPD files for streaming.

Editor’s note: Requirements should be more explicit.

## Content replace and delete functions:

Content management systems may be required to replace, delete, or migrate content assets based on explicit directives from content owners or due to server health monitoring systems. It should allow for approaches to delete manifest(s) when content files cease to exist.

Editor’s note: Requirements should be more explicit.

## Manifest corruption or absence at the server:

Distribution and migration of content assets (content files and manifests) may cause corruption or disappearance of content files or manifests. It should allow for approaches to retrieve either part depending on which part of the content asset is available. Additionally, migration of content assets needs to include a way to recreate an S-MPD if needed.

Editor’s note: Requirements should be more explicit.

## Content annotation:

How much content annotation is expected to be supported by S-MPD? What type of annotations and which part of this information should be repeated in each track if any?

Editor’s note: Content annotation requirements are well described in the DASH content model, which might be used as the starting point.

# References

[1] DASH-IF live media ingest technical specification: <https://dashif.org/guidelines/#dash-if-technical-specification-live-media-ingest>