DASH Industry Forum

Specification of Live Media Ingest

Living Document, 29 March 2019

This version:
https://dashif.org/guidelines/

Issue Tracking:
GitHub

Editors:
DASH IOP Ingest TF

Table of Contents

1. Specification: Live Media Ingest§

1.1. Abstract§

This document presents a Live Media Ingest Protocol specification. Two protocol interfaces are defined. The first, CMAF ingest, is based on fragmented MPEG-4 as defined by the common media application track format (CMAF). The second interface is based on push based MPEG DASH and HLS and may also use the common application track format (CMAF). Both Interfaces use the HTTP POST Method for transmission. Examples of live streaming workflows using these protocol interfaces are included. The protocol also supports carriage of timed metadata and timed text.

1.2. Copyright Notice and Disclaimer§

Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License

This is a document made available by DASH-IF. The technology embodied in this document may involve the use of intellectual property rights, including patents and patent applications owned or controlled by any of the authors or developers of this document. No patent license, either implied or express, is granted to you by this document. DASH-IF has made no search or investigation for such rights and DASH-IF disclaims any duty to do so. The rights and obligations which apply to DASH-IF documents, as such rights and obligations are set forth and defined in the DASH-IF Bylaws and IPR Policy including, but not limited to, patent and other intellectual property license rights and obligations. A copy of the DASH-IF Bylaws and IPR Policy can be obtained at http://dashif.org/.

The material contained herein is provided on an AS IS basis and to the maximum extent permitted by applicable law, this material is provided AS IS, and the authors and developers of this material and DASH-IF hereby disclaim all other warranties and conditions, either express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of accuracy or completeness of responses, of workmanlike effort, and of lack of negligence. In addition, this document may include references to documents and/or technologies controlled by third parties. Those third party documents and technologies may be subject to third party rules and licensing terms. No intellectual property license, either implied or express, to any third party material is granted to you by this document or DASH-IF. DASH-IF makes no any warranty whatsoever for such third party material.

2. Introduction§

The main goal of this specification is to define the interoperability point between live sources (ingest sources) and media processing entities that typically reside in the cloud or the network. This specification does not impose any new constraints or requirements for live streaming to media clients on end-user devices that consume streams using any defined streaming protocol, with a preference for [MPEGDASH]

This document presents protocol interfaces for Live Media Ingest. Live media ingest happens between an ingest source, such as a live video encoder live encoder and distributed media processing entities that receive the ingest stream. Examples of such media processing entities include media packagers, streaming origins and content delivery networks. The structure setup by these media processing entities for receiving the ingest is sometimes referred to as a Publishing point. The combination of ingest sources and distributed media processing entities is common in practical video streaming deployments. In such deployments, interoperability between ingest sources and downstream processing entities can sometimes be challenging. This challenge comes from the fact that each vendor has a different view of what is expected/preferred as well as how various technical specifications apply.

For example, the network protocol for transmission of data and the setup of the connectivity are important. This includes schemes for establishing the ingest connection, handling disconnects and failures, providing procedures for reliabilty sending and receiving the data, and timely resolution of hostnames.

A second level of interoperability lies in the media container and coded media formats. The Moving Picture Experts Group defined several media container formats such as [ISOBMFF] and [MPEG2TS] which are widely adopted and well supported. However, these are general purpose formats, targeting several different application areas. To do so, they provide many different profiles and options. Detailed interoperability is often achieved through other application standards such as those for the broadcast or VOD domains. For interoperable live media ingest, this document provides guidance on how to use [ISOBMFF] and [MPEGCMAF].

In addition, the codec and profile used, e.g. [MPEGHEVC] are important interoperability points that itself also have different profiles and different configurations. This specification provides some guidance on how encoded media should be represented and transmitted.

A third level of interoperability, lies in the way metadata is inserted in streams. Live content often needs methods to signal opportunities for ad insertion or other attributes like timed graphics or general information relating to the broadcast. Examples of such metadata include [SCTE35] markers which are often found in broadcast streams and ID3 tags [ID3v2] for timed events. In fact, many more types of metadata relating to the live event might be ingested and passed on to an OTT workflow.

Fourth, for live media, handling the timeline of the presentation consistently is important. This includes sampling of media, avoiding timeline discontinuities and synchronizing timestamps attached by different ingest sources such as audio and video. In addition, media timeline discontinuities must be avoided as much as possible in normal operation. Further, when using redundant ingest sources, ingested streams must be sample accurately synchronized. Last, streams may need to be started at the same time so as to avoid miss alignment between audio and video tracks.

Fifth, in streaming workflows it is important to have support for failovers of both the ingest sources and media processing entities. This is important to avoid interruptions of 24/7 live services or high profile events where component failure must be expected.

This document attempts to define these interoperability points based on known standardized technologies that have been tested and deployed in several large scale streaming deployments.

3. Conventions and Terminology§

The following terminology is used in the rest of this document.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

ISOBMFF : the ISO Base Media File Format specified in [ISOBMFF].

CMAF Ingest: Ingest interface defined in this specification for push based [MPEGCMAF]

DASH Ingest: Ingest interface defined in this specification for push based [MPEGDASH]

HLS Ingest: Ingest interface defined in this specification for push based [RFC8216]

Ingest Stream: The stream of media pushed from the ingest source to the media processing entity in a live event

Live stream event: The total live stream for the ingest relating to a broadcast event.

Live encoder: Entity performing live encoding of a high quality Ingest stream, can serve as ingest source

Ingest source: A media source ingesting media content to media processing entity , typically a live encoder but not restricted to this, e.g. it could be a stored media resource.

Publishing point : Entry point used to receive an ingest stream, consumes/receives the incoming media ingest stream, typically via a publishing URL setup to receive the stream

Manifest objects Objects ingested that represent streaming manifest e.g. .mpd in MPEG DASH, .m3u8 in HLS

Media objects Objects ingested that represent the media, and or timed text, or other non manifest objects, typically these are CMAF addressable media objects such as CMAF chunks, fragments or segments.

Objects Objects ingested by the ingest source such as manifest objects and media objects (media segments, subtitle segments)

Streaming presentation Manifest objects and media objects composing a Streaming presentation based on a streaming protocol suc h as for example [MPEGDASH]

Media processing entity: Entity used to process the media content, receives/consumes a media [=Ingest stream].

Receiving entity: Entity used to receive the media content, receives/consumes an [=Ingest stream].

CMAFstream : Can be defined using the IETF RFC 5234 ANB [RFC5234] as follows. **CMAFstream** = headerboxes fragments: headerboxes = ftyp moov fragments = X fragment fragment = Moof Mdat

Media fragment Media fragment, combination of moof and mdat in ISOBMFF structure (MovieFragmentBox and mediaDataBox), can be a CMAF fragment or chunk

CMAF Header : CMAF track header defined in [MPEGCMAF]

CMAF Media object : CMAF media object defined in [MPEGCMAF]

CMAF fragment : CMAF fragment defined in [MPEGCMAF]

CMAF chunk : CMAF chunk defined in [MPEGCMAF]

CMAF segment : CMAF segment defined in [MPEGCMAF]

CMAF Track CMAF Track defined in [MPEGCMAF]

HTTP POST : Command used in the Hyper Text Transfer Protocol for sending data from a source to a destination [R

FC7235]

POST_URL : Target URL of a POST command in the HTTP protocol for posting data from a source to a destination.

TCP: Transmission Control Protocol (TCP) as defined in [RFC793]

Arrival Time: The time a metadata item is seen/observed by the application for the first time, e.g. an announcement/avail. The time the event is received (event received time)

Application time : The time a metadata event is applied to a stream (if applicable), correspond to the presentation_time of a dash event [MPEGDASH] (event presentation time)

Connection: A connection setup between two hosts, typically the media ingest source and media processing entity.

Switching set: Group of tracks corresponding to a switching set defined in [MPEGCMAF] or an adaptationset in [MPEGDASH]

ABR : Adaptive Bit-Rate

RTP : Real Time Protocol

OTT : Over the top transmission (HTTP based video streaming)

moof: The MovieFragmentBox "moof" box as defined in the ISOBMFF base media file format [ISOBMFF] that defines the metadata of a fragment.

ftyp: The FileTypeBox "ftyp" box as defined in the ISOBMFF [ISOBMFF]

moov: The container box for all metadata MovieBox "moov" defined in the ISOBMFF base media file format [ISOBMFF] **mdat** : The mediaDataBox "mdat" box defined in ISOBMFF [ISOBMFF].

mfra: The movieFragmentRandomAccessBox "mfra" box defined in the ISOBMFF [ISOBMFF] to signal random access samples (these are samples that require no prior or other samples for decoding) [ISOBMFF].

tfdt : The TrackFragmentBaseMediaDecodeTimeBox box "tfdt" defined in [ISOBMFF] used to signal the decode time of the media fragment signalled in the moof box.

basemediadecodetime : Decode time of first sample as signalled in the tfdt box

mdhd : The MediaHeaderBox "mdhd" as defined in [ISOBMFF], this box contains information about the media such as timescale, duration, language using ISO 639-2/T [iso-639-2] codes [ISOBMFF]

elng : Extended language tag box "elng" defined in [ISOBMFF] that can override the language information

nmhd : The nullMediaHeaderBox "nmhd" as defined in [ISOBMFF] to signal a track for which no specific media header is defined, used for metadata tracks

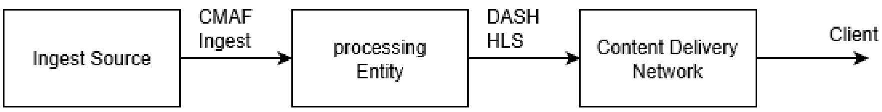
4. Media Ingest Workflows and Profiles§

Two key workflows have been identified for which supporting media ingest interfaces are defined. The two interfaces share a common protocol and may share a common media format. These interfaces are presented separately to make it easier on the implementator by having all relevant information within one section of the document.

The first workflow, CMAF Ingest (Common Media Application Format[MPEGCMAF]), uses CMAF as the encoded media format for contribution to an active media processing entity. The CMAF format is supported by both MPEG-DASH and HLS making it an ideal candidate as a contribution format for preparing Media content for Over the Top delivery. In this case the live media is ingested into the media processing entity which can actively manipulate the received media content and perform operations like on-the-fly encryption, content stitching, packaging and possibly other operations before delivery of the final media presentation to the client.

This type of active media processing offloads the ingest source allowing it to focus on its main role - encoding content. As long as the stream originating from the ingest source contains sufficient metadata, the media processing entity can generate the necessaryt media presentation for streaming to clients or other derived media presentations as needed. Diagram 1 shows a high-level live media workflow from an ingest source towards a media processing entity. In this example the media processing entity prepares the final media presentation for the client which is then delivered to the client via the Content Delivery Network.

Diagram 1: Example with CMAF Ingest



A second interface referred as DASH and HLS ingest is included for ingest of pre-formatted media streaming presentations to entities where the media is not altered actively. The ingest can be based on DASH Ingest or HLS Ingest and includes sending the respective manifest. manifest objects and Media objects are sent using individual fixed length HTTP POST commands to paths that correspond to paths defined in the manifest. While CMAF ingest can also support such operation, it must be done such that each segment is posted as a individual file rather than a long running post of multiple segments to the same file. The main benefit of such processing entities is their simplicity as it allows the use of readily available Web Servers to act as origins for linear content. For workflows that only require a single end client delivery format (e.g., HLS), using the target client format as the ingest format vastly simplifies the workflow and reduces the potential overhead caused by having to manipulate/reformat the media content to a client friendly format in real-time. Note that CMAF Media objects can also be used per

Diagram 2 shows the example in workflow 2 were content is ingested directly into a Content Delivery Network. The content delivery network enables the delivery to the client.

Diagram 2: Example with DASH Ingest

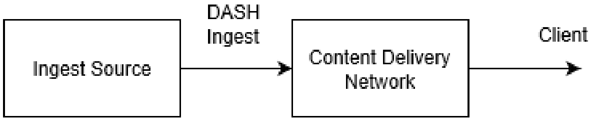


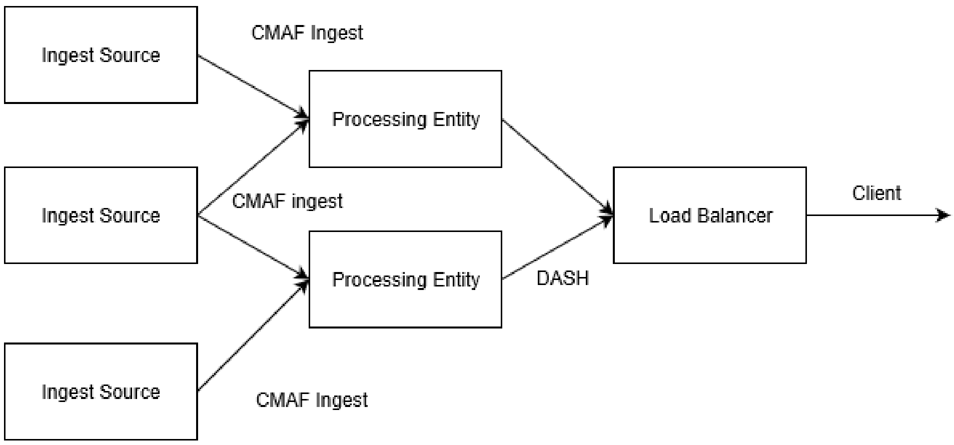
Table 1 highlights some of the key differences and practical considerations of the two interfaces. The best choice for a specific platform depends on many of the use case specific requirements, circumstances and the available technologies.

Table 1: different ingest use cases

Profile	*Ingest source*	*Media processing*
CMAF Ingest	Limited overview, simpler encoder, multiple sources	re-encryption, transcode, stitching, watermark, packaging
DASH/HLS Ingest	Global overview, targets duplicate presentations, limited flexibility no redundancy	manifest manipulation, transmission, storage

Finally, Diagram 3 highlights another aspect that was taken into consideration for large scale systems with many users. Often content owners would like to run multiple ingest sources, multiple receiving entities and make them available to the clients in a seamless fashion for maximum resiliancey. This approach is common when serving web pages, and this architecture also applies to video streaming platforms. In Diagram 3 it is highlighted how one or more Ingest Sources can be sending data to one or more processing entities. In such a workflow it is important to handle the case when one ingest source or media processing entity fails. Both the system and client behavior is an important consideration in practical video streaming systems that need to run 24/7 such as Internet Television. Failovers must be handled robustly and without causing service interruption. This specification details how this failover and redundancy support can be achieved.

Diagram 3: workflow with redundant Ingest sources and receiving entities



5. General Ingest Protocol Behavior§

The media ingest follows the following general requirements for both target /interfaces.

1. The ingest source SHALL communicate using the HTTP POST method as defined in the HTTP protocol, version 1.1 [RFC7235]
2. The ingest source SHOULD use HTTP over TLS, if TLS is used it SHALL be TLS version 1.2 or higher [RFC2818]
3. The ingest source SHOULD repeatedly resolve the hostname to adapt to changes in the IP to Hostname mapping such as for example by using the domain naming system DNS [RFC1035] or any other system that is in place.
4. The ingest source MUST update the IP to hostname resolution respecting the TTL (time to live) from DNS query responses, this will enable better resilience to changes of the IP address in large scale deployments where the IP address of the media processing entities may change frequently.
5. In case HTTP over TLS [RFC2818] protocol is used, basic authentication HTTP AUTH [RFC7617] or TLS client certificates MUST be supported.
6. Mutual authentication SHALL be supported. Client certificates SHALL chain to a trusted CA , or be self assigned.
7. As compatibility profile for the TLS encryption the ingest source SHOULD use the mozzilla intermediate compatibility profile MozillaTLS.
8. In case of an authentication error, the ingest source SHALL retry establishing the Connection within a fixed time period with updated authentication credentials
9. The ingest source SHOULD terminate the HTTP POST request if data is not being sent at a rate commensurate with the MP4 fragment duration. An HTTP POST request that does not send data can prevent media processing entities from quickly disconnecting from the ingest source in the event of a service update.
10. The HTTP POST for sparse data SHOULD be short-lived, terminating as soon as the data of a fragment is sent.
11. The POST request uses a POST_URL to the basepath of the publishing point at the media processing entity and SHOULD use an additional relative path when posting different streams and fragments, for example, to signal the stream or fragment name.

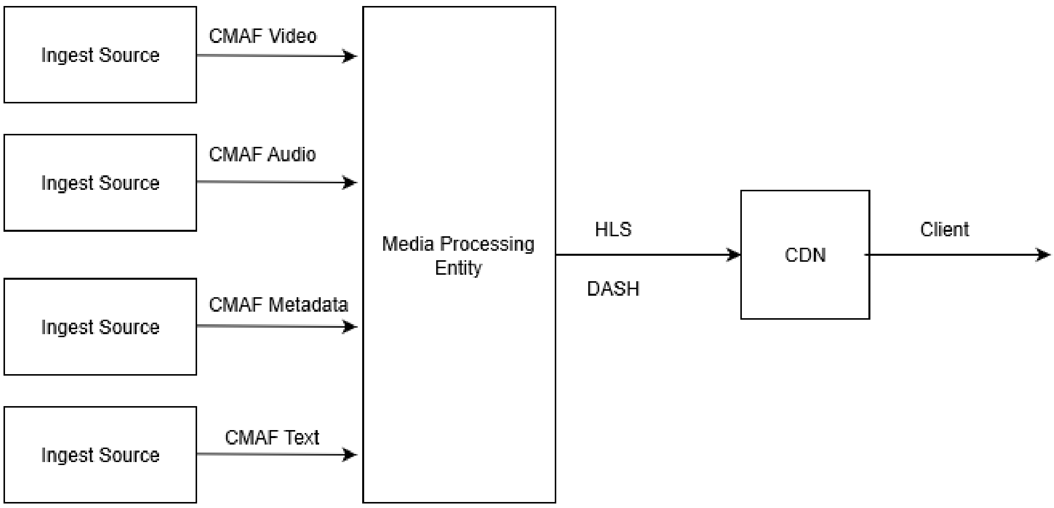
6. CMAF Ingest General Considerations§

The binary media format for conveying the media is based on CMAF track constraints as specified in [MPEGCMAF]. A key benefit of this format is that it allows easy identification of stream boundaries, enabling switching, redundancy, re-transmission resulting in a good fit with the current Internet infrastructures. Many problems in practical streaming deployment often deal with issues related to the binary media format.

We believe that the CMAF track format will make things easier and that the industry is already heading in this direction following recent specifications like [MPEGCMAF] and HLS [RFC8216].

CMAF ingest assumes ingest to an active media processing entity, or any other entity such as a storage or origin server, from one or more Ingest source, ingesting one or more types of media streams. This advances over the ingest part of the smooth ingest protocol MS-SSTR by only using standardized media container formats and boxes based on [ISOBMFF] and [MPEGCMAF]. In addition, this allows extension towards codecs like [MPEGHEVC] and timed metadata ingest of subtitle and timed text streams. The workflow ingesting multiple media ingest streams with fragmented MPEG-4 ingest is illustrated in Diagram 6. Discussions on the early development have been documented fmp4git.

Diagram 6: fragmented MPEG-4 ingest with multiple ingest sources



Diagrams 7-9 detail some of the concepts and structures. Diagram 7 shows the data format structure of the CMAF Track format [ISOBMFF] and [MPEGCMAF]. In this format media meta data such as playback time, sample duration and sample data (encoded samples) are interleaved. The MovieFragmentBox moof box as specified in [ISOBMFF] is used to signal the information to playback and decode the samples stored in the following mdat box. The ftyp and moov box contain the track specific information and can be seen as a CMAF Header of the stream, sometimes referred as a [MPEGCMAF] header. The combination of moof mdat can be referred as a CMAF fragment or CMAF chunk or a CMAF segment depending on the structure content and the number of moof mdat structures in the addressable object.

The combination of ftyp and moov can be referred to as a CMAF header. These CMAF Addressable media objects can be jointly referred to as CMAF Media object

Diagram 7: CMAF Track stream:

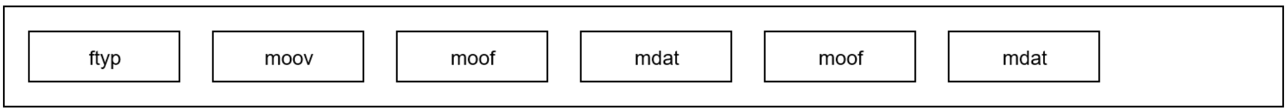


Diagram 8 illustrates the synchronization model, that is based on the synchronization model proposed in [MPEGCM

AF]. Different bit-rate tracks and/or media streams are conveyed in separate CMAF tracks. By having the boundaries to the fragments time aligned for tracks comprising the same content stream at different bit-rates, bit-rate switching can be achieved. By using a common timeline different streams can be synchronized at the receiver, while they are in a separate CMAF Track, send over a separate connection, possibly from a different Ingest source. For more information on the synchronization model we refer to section 6 of [MPEGCMAF]. For synchronization of tracks coming from different encoders, sample time accuracy is required. i.e. the same samples need to be mapped to the sample time on the timescale used for the track. Further, in case multiple redundant ingest sources are used they must present sample accurately synchronized streams.

In diagram 9 another advantage of this synchronization model is illustrated, the concept of late binding. In the case of late binding, a new stream becomes available and is adopted later in a presentation. By using the fragment boundaries and a common timeline it can be received by the media processing entity and embedded in the presentation. Late binding is useful for many practical use cases when broadcasting television content with different types of media and metadata tracks originating from different sources.

Note that it is important, as defined in MPEG CMAF that different CMAF Tracks have the same starting time sharing an implicit timeline. A stream becoming available late needs to be synchronized and time aligned with other streams ingested avoiding miss alignment and other issues.

Diagram 8: CMAF Track synchronization:

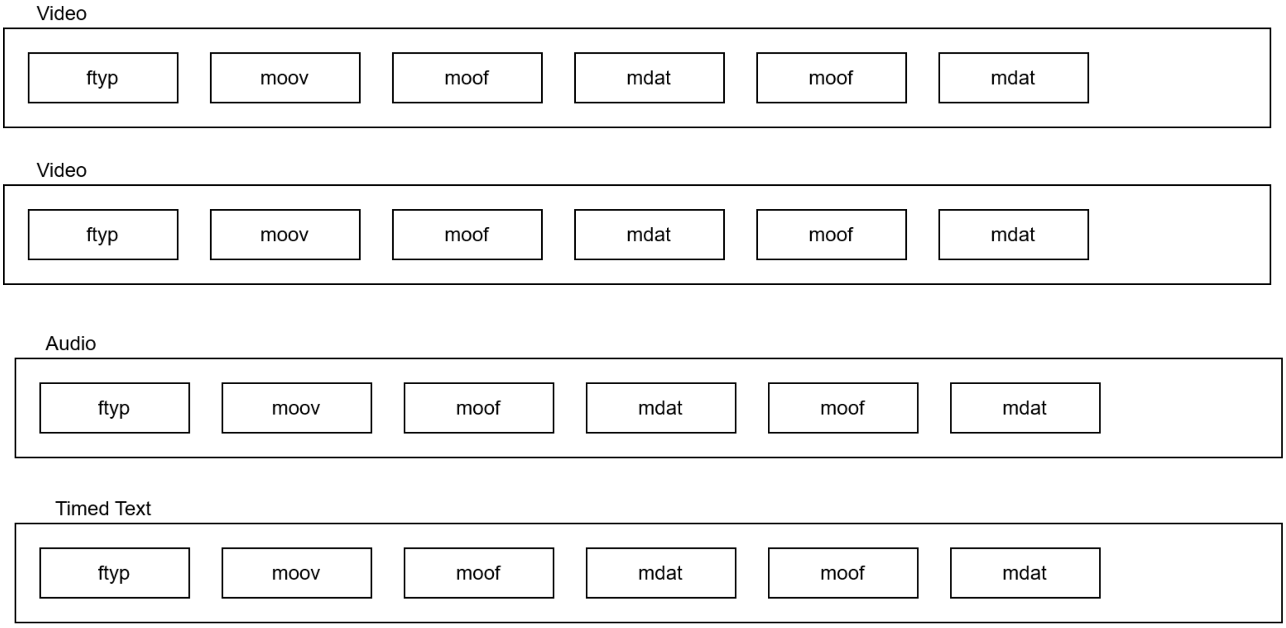


Diagram 9: CMAF late binding:

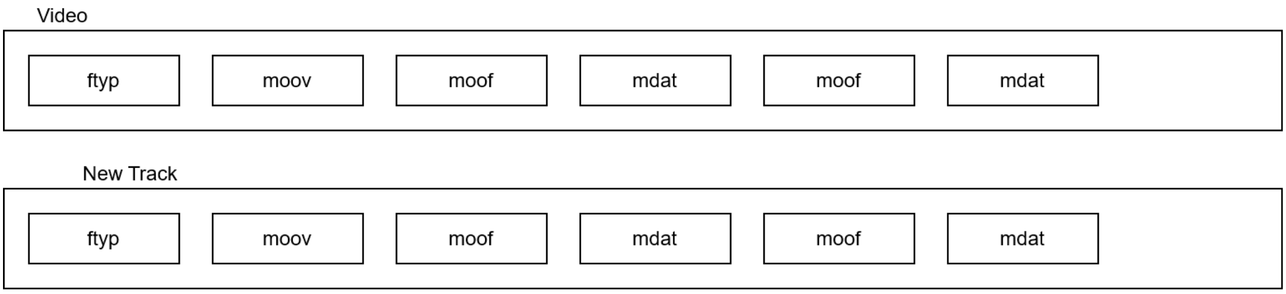
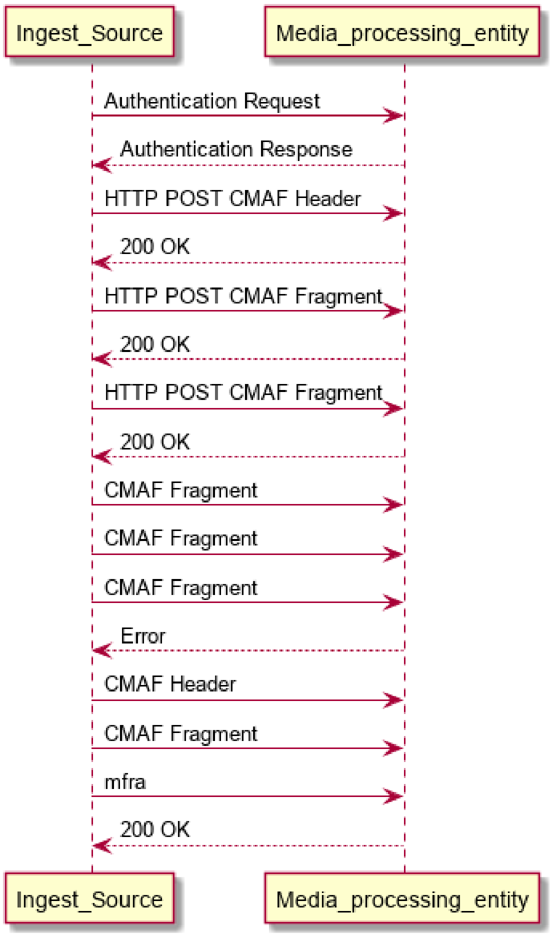


Diagram 10 shows the flow of the media ingest. It starts with a DNS resolution (if needed) and an authentication step (using Authy, or TLS certificates) to establish a secure TCP connection. In some private datacenter deployments where nodes are not reachable from outside, a non authenticated connection may also be used. The ingest source then issues a POST to test that the media processing entity is listening. This POST contains the moov + ftyp box (the init fragment or CMAF Header or could be empty. In case this is successful this is followed by the rest of the fragments in the CMAFstream. At the end of the session, for tear down the source can send an empty mfra box to close the connection. This is then followed with a zero length chunk, allowing the receiver to send a response, the encoder can follow up by closing the TCP connection using a FIN command as defined in HTTP RFC2616.

Diagram 10: CMAF ingest flow



7. Ingest Interface 1: CMAF Ingest Protocol Behavior§

This section describes the protocol behavior specific to interface 1: CMAF ingest. Operation of this profile MUST also adhere to the general requirements.

7.1. General Protocol Requirements§

1. The ingest source SHALL start by sending an HTTP POST request with the CMAF Header, or an empty request, by using the POSTURL This can help the ingest source to quickly detect whether the publishing point is valid, and if there are any authentication or other conditions required.
2. The ingest source MUST initiate a media ingest connection by posting the CMAF header after step 1
3. The ingest source SHOULD use the chunked transfer encoding option of the HTTP POST command [RFC2626] when the content length is unknown at the start of transmission or to support use cases that require low latency
4. If the HTTP POST request terminates or times out with a TCP error, the ingest source MUST establish a new connection, and follow the preceding requirements. Additionally, the ingest source MAY resend the fragment in which the timeout or TCP error occurred.
5. The ingest source MUST handle any error responses received from the media processing entity, by establishing

a new connection and following the preceding requirements including retransmitting the ftyp and moov boxes or the CMAF Header.

6. In case the Live stream event is over the ingest source SHALL signal the stop by transmitting an empty mfra box towards the media processing entity. After that it SHALL send an empty HTTP chunk, Wait for the HTTP response before closing TCP session RFC2616 when this response is received
7. The Ingest source SHOULD use a separate TCP connection for ingest of each different CMAF track
8. The Ingest source MAY use a separate relative path in the POST_URL for ingesting each different track by appending it to the POST_URL, this can make it easy to detect redundant streams from different ingest sources.
9. The base media decode timestamps basemediadecodetime in tfdt of fragments in the CMAFstream SHOULD arrive in increasing order for each of the fragments in the different tracks/streams that are ingested.
10. The fragment sequence numbers seq_num of fragments in the CMAFstream signalled in the tfhd SHOULD arrive in increasing order for each of the different tracks/streams that are ingested. Using both timestamp basemediadecodetime and seq_num based indexing will help the media processing entities identify discontinuities in the ingest stream.
11. Stream names MAY be signalled by adding the relative path Stream(stream_name) to the POST_URL, this can be useful for identification when multiple ingest sources send the same redundant stream to a receiver
12. The average and maximum bitrate of each track SHOULD be signalled in the btrt box in the sample entry of the CMAF header or init fragment
13. In case a track is part of a Switching set, all properties section 6.4 and 7.3.4 of [MPEGCMAF] MUST be satisfied, enabling the receiver to group the tracks in respective switching sets
14. Ingested tracks MUST conform to CMAF track structure defined in [MPEGCMAF]
15. CMAF Tracks SHOULD NOT use segmentTypeBox to signal CMAF Media object brands like chunk, fragment, segment.

7.2. Requirements for Formatting Media Tracks§

1. Media tracks SHALL be formatted using boxes according to section 7 of [MPEGCMAF] except for section 7.4. which dictates boxes that are not compliant to [ISOBMFF] relating to encryption and DRM systems
2. The trackFragmentDecodeTime box tfdt box MUST be present for each fragment posted.
3. The ISOBMFF media fragment duration SHOULD be constant, the duration MAY fluctuate to compensate for non-integer frame rates. By choosing an appropriate timescale (a multiple of the frame rate is recommended) this issue should be avoided.
4. The fragment durations SHOULD be between approximately 1 and 6 seconds.
5. The CMAF Tracks SHOULD use a timescale for video streams based on the framerate and 44.1 KHz or 48 KHz for audio streams or any another timescale that enables integer increments of the decode times of fragments signalled in the "tfdt" box based on this scale. If necessary, integer multiples of these timescales could be used.
6. The language of the CMAF Track SHOULD be signalled in the mdhd box or elng boxes in the init fragment, cmaf header and/or moov headers (mdhd).
7. Media CMAF tracks SHOULD contain the bitrate btrt box specifying the target average and maximum bitrate of the fragments in the sample entry container in the init fragment/CMAF header
8. The CMAF track MAY comprise CMAF chunks [MPEGCMAF] which are moof mdat structures that may not be an IDR or switching point
9. For video tracks, profiles like avc1 and hvc1 MAY be used that signal the sequence parameter set in the CMAF Header in the sample entry. In this case parameters do not change dynamically during the live event and are signalled in the moviebox of the CMAF Header.

10. Alternatively, video tracks MAY use profiles like avc3 or hev1 that signal the parameter sets (PPS, SPS, VPS) in in the media samples.
11. In case the language of track changes a new init fragment with update mdhd and or elng SHOULD be send.
12. Track roles can be signalled in the ingest by using a kind box in userData udta box. The kind box MUST contain a schemeIdUri MPEG urn:mpeg:dash:role:2011 and a value containing a Role as defined in [MPEGDASH]

Note: [MPEGCMAF] has the notion of a segment, a fragment and a chunk. A fragment can be composed of one or more chunks, while a segment can be composed of one or more fragments. The Media fragment defined here is independent of this notion and can be a chunk, a fragment containing a single chunk or a segment containing a single fragment containing a single chunk. In this text we use Media fragment to denote the structure combination moof mdat.

7.3. Requirements for Signalling Switching Sets§

In live streaming a bundle of streams corresponding to a channel is ingested by posting to a publishing point. CMAF has the notion of switchingsets [MPEGCMAF] which map to similar streaming protocol concepts like adaptationset in [MPEGDASH]. To signal a switching set CMAF media tracks MUST correspond to the constraints defined in [MPEGCMAF] section 7.3.4 . Table 2 summarizes the CMAF Switching set constraints.

Table 2: Switching set constraints

Box	*General CMAF header constraints in a CMAF switching set*
ftyp	Shall be identical except for media profile brands (see 1 in 7.3.4.1)
mvhd	Shall be identical except for creation_time, and modification_time
tkhd	Shall be identical except for width, height, creation_time, and modification_time. See NOTE 1.
trex	identical
elst	Shall be identical except for video CMAF track files with a different composition offset
mdhd	Shall be identical except for creation_time, and modification_time
mehd	Global overview, targets duplicate presentations
meta	May contain different boxes and data
udta	May contain different boxes and data
cprt	identical
kind	identical
elng	identical
hdlr	identical
vmhd	identical
smhd	identical
sthd	identical
dref	identical

stsd	Sample entries shall have the same codingname (four-character code)

NOTE 1: Track width and height can differ, but picture aspect ratio is the same for all CMAF tracks. NOTE 2 Sample entry constraints for CMAF switching sets are defined by each CMAF media profile

For additional signalling of CMAF tracks belonging to the same switching set, the ingest source MAY set the alternate_group value in the TrackHeaderBox tkhd to a value that is the same for tracks belonging to the same switching set. This allows explicit signalling of tracks that do apply to switchingset constraints but do not belong to the same switching set. Alternatively one could signal switching explicitly by means outside of this specification.

7.4. Requirements for Timed Text, Captions and Subtitle Streams§

The live media ingest specification follows requirements for ingesting a track with timed text, captions and/or subtitle streams. The recommendations for formatting subtitle and timed text track are defined in [MPEGCMAF] and [MPEG 4-30] and are re-iterated here for convenience to the reader. Note that the text in [MPEGCMAF] prevails the text below when different except for the notion of 9 and 10-11 on roles adding a bitrate box.

1. The track SHOULD be a sparse track signalled by a null media header nmhd containing the timed text, images, captions corresponding to the recommendation of storing tracks in CMAF [MPEGCMAF], or a sthd for an ISOBMFF subtitle track (e.g. TTML)
2. Based on this recommendation, the trackhandler "hdlr" SHALL be set to "text" for WebVTT and "subt" for TTML following [MPEG4-30]
3. In case TTML is used the track MUST use the XMLSampleEntry to signal sample description of the sub-title stream [MPEG4-30]
4. In case WebVTT is used the track must use the WVTTSampleEntry to signal sample description of the text stream [MPEG4-30]
5. These boxes SHOULD signal the mime type and specifics as described in [MPEGCMAF] sections 11.3 ,11.4 and 11.5
6. The boxes described in 2-4 must be present in the init fragment (ftyp + moov) or cmaf header for the given track
7. subtitles in CTA-608 and CTA-708 format SHOULD be conveyed following the recommendation section 11.5 in [MPEGCMAF] via Supplemental Enhancement Information SEI messages in the video track [MPEGCMAF]
8. The ftyp box in the CMAF Header for the track containing timed text, images, captions and sub-titles MAY use signalling using CMAF profiles based on [MPEGCMAF]

   8a. WebVTT Specified in 11.2 ISO/IEC 14496-30 [MPEG4-30] *cwvt*

   8b.TTML IMSC1 Text Specified in 11.3.3 [MPEG4-30] IMSC1 Text Profile *im1t*

   8c.TTML IMSC1 Image Specified in 11.3.4 [MPEG4-30] IMSC1 Image Profile *im1i*

   8d. CEA CTA-608 and CTA-708 Specified in 11.4 [MPEG4-30] Caption data is embedded in SEI messages in video track ccea
9. The BitRateBox btrt SHOULD be used to signal the average and maximum bitrate in the sample entry box, this is most relevant for bitmap or xml based timed text subtitles that may consume significant bandwidths (e.g. im1i)
10. In case the language of a track changes, a new init fragment or CMAF Header with updated mdhd and/or elng SHOULD be send from the ingest source to the media processing entity.
11. Track roles can be signalled in the ingest, by using a kind box in udta box. The kind box MUST contain a schemeIdUri MPEG urn:mpeg:dash:role:2011 and a value containing a Role as defined in [MPEGDASH]

Note: [MPEGCMAF] allows multiple kind boxes, hence multiple roles can be signalled. By default one should signal the DASH role urn:mpeg:dash:role:2011. A receiver can derive corresponding configuration for other streaming protocols such as HLS [RFC8216]. In case this is not desired, additional kind boxes with corresponding schemeIdUri and values can be used to explicitly signal this kind of information. Subschemes can be signalled in the schemeIdURI as schemeIdURI@value.

An informative scheme of defined roles in MPEG DASH and respective corresponding roles in HLS [RFC8216] can be found below, additionally the forced subtitle in HLS might be derived from a DASH forced subtitle role

Table 3: Roles for subtitle and Audio tracks and HLS Characteristics

Characteristic [RFC8216]	*urn:mpeg:dash:role:2011*
transcribes-spoken-dialog	subitle
easy-to-read	easyreader
description	description

MPEG DASH roles are defined in urn:mpeg:dash:role:2011 [MPEGDASH]. Additionally another example for explicitly signalling roles could b e DVB DASH [DVB-DASH]. One could use schemeiduri@value and role as defined there. e.g. kind.schemeIdUri="urn:tva:metadata:cs:AudioPurposeCS:2007@1 kind.value=Alternate

7.5. Requirements for Timed Metadata§

This section discusses the specific formatting requirements for CMAF ingest of timed metadata related to events and markers for ad insertion or other timed metadata. An example of these are opportunities for splice points and program information signalled by SCTE-35 markers. This type of event signalling is different from regular audio/video information because of its sparse nature. In this case, the signalling data usually does not happen continuously, and the intervals can be hard to predict.

Examples of timed metadata are ID3 tags [ID3v2], SCTE-35 markers [SCTE35] and DASH emsg messages defined in section 5.10.3.3 of [MPEGDASH]. In addition, any other metadata can be signalled in this scheme by providing a URI to identify the scheme, and the metadata embedded as samples in mdat. For example, DASH Event messages contain a schemeIdUri that defines the payload of the message.

Table 4 provides some example urn schemes to be signalled in the emsg Table 5 illustrates an example of a SCTE-35 marker stored in a DASH emsg.

The presented approach enables ingest of timed metadata from different sources, possibly on different locations by embedding them in sparse metadata tracks. In this approach metadata are not interleaved with the media as for example the case in emsg boxes in [MPEGCMAF]. However, by embedding the emsg structure as samples the benefits of its usages in DASH and CMAF are kept.

Example metadata messages include inband event message box as used in [MPEGDASH], [DVB-DASH], or alternatively direct embedding of [SCTE35] or [ID3v2] which might in some cases be used.

Table 4: Roles for subtitle and Audio tracks and HLS Characteristics

SchemeIdURI	*Reference*
urn:mpeg:dash:event:2012	[MPEGDASH], 5.10.4 subtitle
urn:dvb:iptv:cpm:2014	[DVB-DASH], 9.1.2.1

urn:scte:scte35:2013:bin	[SCTE35] 14-3 (2015), 7.3.2
www.nielsen.com:id3:v1	Nielsen ID3 in MPEG-DASH [ID3v2]

Table 5: Example of a SCTE-35 marker embedded in a DASH eventmessagebox

Tag	*Value*
scheme_uri_id	urn:scte:scte35:2013:bin
Value	value used to signal subscheme
Timescale	positive number, ticks per second, similar to track timescale
presentation_time_delta	non-negative number, splice time compared to tfdt
event_duration	duration of event "0xFFFFFFFF" if unknown
id	unique identifier for message
message_data	splice info section including CRC

Alternatively, a version 1 of the eventmessagebox with absolute timing could be used, where the presentation time is added as a 64 bit integer. In this case care must be taken not to signal events in the past or too far in the future.

The following steps are recommended for timed metadata ingest related to events, tags, ad markers and program information:

1. Metadata SHALL be conveyed in a CMAF track, where the media handler (hdlr) is "meta", the track handler box is a null media header box nmhd.
2. The metadata track applies to the media streams ingested to a Publishing point entry at the media processing entity or origin server
3. The URIMetaSampleEntry entry SHALL contain, in a URIbox, the URI following the URI syntax in [RFC3986] defining the form of the metadata (see the ISO Base media file format specification [ISOBMFF]).
4. The URIMetaSampleEntry SHOULD contain the urn urn:mpeg:dash:event:2012 or an equivalent urn to signal the presence of event message boxes
5. The timescale of the metadata SHOULD match the value specified in the media header box "mdhd" of the metadata track.
6. The Arrival time is signalled in the "tfdt" box of the track fragment as the basemediadecode time, this is the time when the metadata will be first received.
7. The Application time can be signalled as a difference to the arrival time by an empty sample with duration delta, the application time is the time when the metadata or event is applied. It is equal to the media presentation time of the sample containing the event/metadata. Alternatively composition time offset can be used to signal the difference between the Arrival and application time.
8. The duration of the sample signalled in the trun box SHOULD correspond to the duration of the metadata if the metadata is valid for a duration of time (if applicable), however, sometimes this is not the case and alternative durations can be used.
9. Empty samples, and fragments with empty samples SHOULD be used to fill the timeline to avoid timeline gaps or 32 bit duration overflow for large timescales
10. All Timed Metadata samples SHOULD be sync samples [ISOBMFF], defining the entire set of metadata for the time interval they cover. Hence, the sync sample table box SHOULD not be present.
11. The payload is conveyed in the mdat box as sample information.

12. In some cases, the duration of the metadata may not be known, in this case the sample duration could be set to zero and updated later when the timestamp of the next metadata fragment is received.
13. The ingest source SHALL not embed inband event message boxes emsg in the ingest stream

> Note: [MPEGCMAF] has the notion of an inband event message box to convey metadata and event messages. In the current specification a separate track is used instead to convey such information. Advantages include avoiding sending duplicate information in multiple tracks, and avoiding a strong dependency between media and metadata by interleaving them. The ingest source shall NOT send inband emsg box and the receiver SHALL ignore it. However, event message box can be embedded as samples in the timed metadata track.

7.6. Requirements for Media Processing Entity Failover§

Given the nature of live streaming, good failover support is critical for ensuring the availability of the service. Typically, media services are designed to handle various types of failures, including network errors, server errors, and storage issues. When used in conjunction with proper failover logic from the ingest sources side, highly reliable live streaming setups can be build. In this section, we discuss requirements for failover scenarios. The following steps are required for an ingest source to deal with a failing media processing entity.

The CMAF ingest source may implement the following recommendations to achieve failover support.

1. The ingest source MUST use a timeout for establishing the TCP connection. If an attempt to establish the connection takes longer, abort the operation and try again.
2. The ingest source MUST resend media fragments for which a connection was terminated early
3. The ingest source SHOULD NOT limit the number of retries to establish a connection or resume streaming after a TCP error occurs.
4. After a TCP error: a. The current connection MUST be closed, and a new connection MUST be created for a new HTTP POST request. b. The new HTTP POST URL MUST be the same as the initial POST URL for the fragment to be ingested. c. The new HTTP POST MUST include stream headers (ftyp, and moov boxes) identical to the stream headers.
5. In case the media processing entity cannot process the POST request due to authentication or permission problems then it SHOULD return a permission denied HTTP 403
6. In case the media processing entity can process the request it SHOULD return an HTTP 200 OK or 202 Accepted
7. In case the media processing entity can process the fragment in the POST request body but finds the media type cannot be supported it SHOULD return an HTTP 415 unsupported media type
8. In case an unknown error happened during the processing of the HTTP POST request a HTTP 404 Bad request SHOULD be returned by the media processing entity
9. In case the media processing entity cannot process a fragment posted due to missing or incorrect init fragment, an HTTP 412 unfulfilled condition SHOULD be returned
10. In case an ingest source receives an HTTP 412 response, it SHALL resend ftyp and moov boxes

7.7. Requirements for Live Media Source Failover§

Live encoder or Ingest source failover is the second type of failover scenario that needs to be supported for end-to-end live streaming delivery. In this scenario, the error condition occurs on the ingest source side. The following expectations apply to the live ingestion endpoint when encoder failover happens:

1. A new ingest source instance SHOULD be instantiated to continue the ingest
2. The ingest source MUST use the same URL for HTTP POST requests as the failed instance.

3. The new ingest source POST request MUST include the same CMAF Header or init fragment as the failed instance
4. The ingest source MUST be properly synced with all other running ingest sources for the same live presentation to generate synced audio/video samples with aligned fragment boundaries. This implies that UTC timestamps for fragments in the "tfdt" match between decoders, and encoders. In addition, fragment boundaries need to be appropriately synchronized.
5. The new stream MUST be semantically equivalent with the previous stream, and interchangeable at the header and media fragment levels.
6. The new instance of ingest source SHOULD try to minimize data loss. The basemediadecodetime tfdt of media fragments SHOULD increase from the point where the encoder last stopped. The basemediadecodetime in the tfdt box SHOULD increase in a continuous manner, but it is permissible to introduce a discontinuity, if necessary. Media processing entities can ignore fragments that it has already received and processed, so it is better to error on the side of resending fragments than to introduce discontinuities in the media timeline.

8. Ingest Interface 2: DASH and HLS Ingest Protocol Behavior§

DASH/HLS is designed to ingest a Streaming presentation composed of Manifest objects and Media objects to receiving entities that provide either pass-through functionality or limited processing of the content. In this mode, the Ingest source prepares and ingests all the Objects intended for consumption by a client. These are complete Streaming presentation including all manifest and media objects.

The requirements below encapsulate all needed functionality to support Interface 2. The requirements listed for Profile 1 in section §7.1 General Protocol Requirements do not apply to Interface 2. General shared requirements are covered in section §5 General Ingest Protocol Behavior. In case [!MPEGCMAF] media is used, the media track and segment formatting will be similar as defined in Interface 1.

8.1. General requirements§

8.1.1. Industry Compliance§

1. The Streaming presentation ingested MUST be either MPEG DASH [MPEGDASH] or HTTP live Streaming [RFC8216] conforming.
2. The ingest source MUST support the use of fully qualified domain names to identify the Receiving entity.
3. Both the ingest source and Receiving entity MUST support IPv4 and IPv6 transport.
4. The ingest source MUST have the capability of specifying the publishing path (which will be used to publish the content) as well as the delivery path (which clients will use to retrieve the content).

These capabilities are further illustrated in the Examples sections, and may be defined outside the scope of this specification.

8.1.2. HTTP connections§

1. Manifest objects and Media objects MUST be uploaded via individual HTTP 1.1 [RFC7235] PUT or POST operations. This specification does not imply any functional differentiation between a PUT or a POST operation. Either may be used to supply content to the receiving entity.
2. Media objects that are not referenced in corresponding Manifest objects SHOULD be removed by the ingest source via an HTTP DELETE operation. A DELETE request should support: 3.1. deleting an empty folder. 3.2. deleting the corresponding folder if the last file in the folder is deleted and it is not a root folder but not necessarily recursively deleting empty folders.
3. Persistent TCP connections SHOULD be used.

4. Multiple Parallel connections SHOULD be used to ingest the streaming presentation that is being concurrently generated. For example, parallel connections can be used for Media objects for different bitrates.
5. If the content length of an object is not known at the start of the upload, for example with low latency chunked encoding, then HTTP 1.1 Chunked transfer encoding MUST be used.

8.1.3. Unique segment and manifest naming§

1. All Media objects (video segments, audio segments, init segments and caption segments) MUST carry unique path names. This uniqueness applies across all ingested content in previous sessions, as well as the current session.
2. All objects in a Live stream event MUST be contained within a root path assigned to it.
3. Manifest objects MUST carry paths which are unique to each live stream event. One suggested method of achieving this is to introduce the timestamp of the start of the live stream event in to the manifest path.
4. Objects uploaded with the same path as a prior object will replace the prior object.
5. Media object names MUST end with a number which is monotonically increasing. It MUST be possible to retrieve this numeric suffix via a regular expression
6. Media objects containing initialization fragments MUST be identified through either using the .init file extension OR " init" in their file name. 'All other objects which do not contain initialization fragments MUST NOT include the string "init" in their file name.
7. All objects must carry a file extension and a MIME-type. The following file extensions and mime-types are the ONLY permissible combinations to be used:

Table 6:

File Extension	*Mime Type*
.m3u8	*application/x-mpegURL or vnd.apple.mpegURL*
.mpd	*application/x-mpegURL*
.cmfv	*video/mp4*
.cmfa	*audio/mp4*
.cmft	*application/mp4*
.cmfm	*application/mp4*
.mp4	*video/mp4 or application/mp4*
.m4v	*video/mp4*
.m4a	*audio/mp4*
.m4s	*video/iso.segment*
.init	*video/mp4*
.header	*video/mp4*
.key	*to be defined*

8.1.4. DNS lookups§

DNS lookup requirements are defined in the general protocol requirements section §5 General Ingest Protocol Behavior.

8.1.5. Ingest source identification§

1. The ingest source MUST include a User-Agent header (which provides information about brand name, version number, and build number in a readable format) in all post messages.

8.1.6. Common Failure behaviors§

The following items define the behavior of an ingest source when encountering certain conditions.

1. When the ingest source receives a TCP connection attempt timeout, abort midstream, response timeout, TCP send/receive timeout or 5xx response when attempting to POST content to the Receiving entity, it MUST a. For manifest objects: re-resolve DNS on each retry (per the DNS TTL) and retry indefinitely. b. For media objects: re-resolve DNS on each retry (per the DNS TTL) and continue uploading for n seconds, where n is the segment duration. After it reaches the media object duration value, drop the current data and continue with the next media object. To maintain continuity of the time-line, the ingest source SHOULD continue to upload the missing media object with a lower priority. Once a media object is successfully uploaded, update the corresponding manifest object with a discontinuity marker appropriate for the protocol format at hand.
2. HTTP 403 or 400 errors For all objects (manifest and non-manifest), do not retry. The ingest source MUST stop ingesting objects and provide a log or fatal error condition.

8.2. HLS specific requirements§

8.2.1. File extensions and mime-types§

1. The parent and child playlists MUST use a .m3u8 file extension.
2. The keyfile, if required, MUST use a .key file extension, if statically served.
3. If segments are encapsulated using a Transport Stream File Format, they MUST carry a ".ts" file extension.
4. If segments are encapsulated using [MPEGCMAF], then they MUST NOT use a ".ts" file extension and must use one of the other allowed file extensions appropriate for the mime-type of the content they are carrying.

8.2.2. Upload order§

In accordance with the HTTP live Streaming [RFC8216] recommendation, ingest sources MUST upload all required files for a specific bitrate and segment before proceeding to the next segment. For example, for a bitrate that has segments and a playlist that updates every segment and key files, ingest sources should upload the segment file followed by a key file (optional) and the playlist file in serial fashion. The encoder should only move to the next segment after the previous segment has been successfully uploaded or after the segment duration time has elapsed. The order of operation should be: 1.1 Upload media segment, 1.2 Optionally upload key file, if required, 1.3 Upload the .m3u8. If there is a problem with any of the Steps, retry them. Do not proceed to Step 3 until Step 1 succeeds or times out as described in common failure behaviors above. Failed uploads MUST result in a stream manifest Discontinuity per [RFC8216].

8.2.3. Encryption§

1. The ingest source MAY choose to encrypt the media segments and publish the corresponding keyfile to the receiving entity.

8.2.4. Relative paths§

1. Relative URL paths SHOULD be used to address each segment.

8.2.5. Resiliency§

1. When ingesting media objects to multiple receiving entities, the ingest source MUST send identical media objects with identical names
2. To allow resumption of failed sessions and to avoid reuse of previously cached content, the ingest source MUST NOT restart object names or use previously used object names.
3. When multiple ingest sources are used, they MUST use consistent media object names including when reconnecting due to any application or transport error. A common approach is to use epoch time/segment duration as the object name.

8.3. DASH specific requirements§

8.3.1. File extensions and mime-types§

1. The manifest objects MUST use a ".mpd" file extension.
2. Media objects MUST NOT use a ".ts" file extension and must use one of the other allowed file extensions defined in this document.

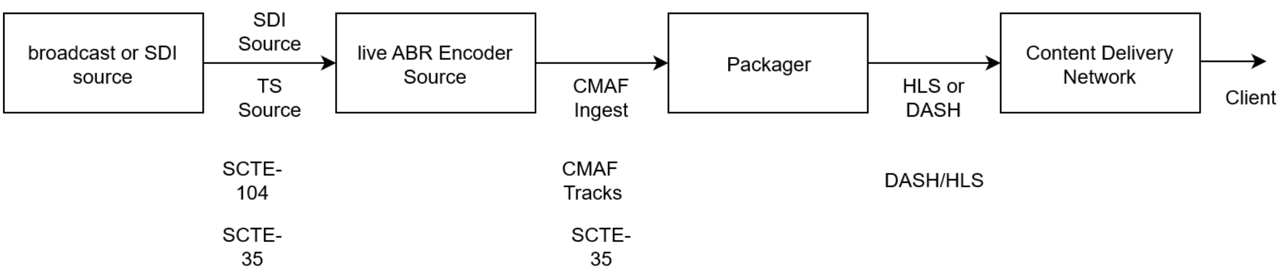
8.3.2. Relative paths§

1. Relative URL paths MUST be used to address each segment.

9. Illustrative Example of using CMAF and DASH ingest specification§

In this section we provide some example deployments for live streaming, mapping to the architecture defined in DASH-IF live Task Force. Diagram 11 shows an example where a separate packager and origin server are used.

Diagram 11: Example setup schema with CMAF ingest and DASH/HLS ingest



The broadcast source is used as input to the live ABR encoder. The broadcast sources can be original SDI signals from a broadcast facility or TS streams intercepted from a broadcast that need to be re-used in an OTT distribution workflow. The live ABR encoder source performs the ABR encoding of the tracks into CMAF tracks and functions as the ingest source in the CMAF ingest interface. Multiple live ABR encoder sources can be used, providing redundant inputs to the packager, which is the media processing entity consuming the CMAF ingest. The packager is receiving the different CMAF tracks. The ingest follows the CMAF Ingest specification in this document, allowing for failover, redundancy and many of the other features related to the content tracks. The live encoder source performs the following tasks:

- It demuxes and receives the MPEG-2 transport stream and/or HD SDI signal
- It formats the metadata in these streams such as SCTE-35 or SCTE 104 to timed metadata tracks
- It performs a high quality ABR encoding in different bit-rates with aligned switching points
- It packages all media and timed text tracks as CMAF compliant tracks and signals track roles in kind boxes
- It POSTs the addressable media objects composing the tracks to the live packager according to the CMAF ingest specification interface defined in this document.
- The CMAF ingest allows multiple live encoder sources and packagers to be deployed benefiting from redundant stream creation, avoiding timeline discontinuities due to failures as much as possible.
- In case the receiver fails, it will reconnect and resend as defined in the section on failover once it reconnects
- In case the live encoder source fails it will restart and perform the steps as detailed in the section on failover

The live encoder source can be deployed in the cloud or on a bare metal server or even as a dedicated hardware. The live encoder source may have some tools or configuration API's to author the CMAF tracks and feed instruction/properties from the original SDI or broadcast into the CMAF tracks. The packager receives the ingested streams, and performs the following tasks.

- It receives the CMAF tracks, grouping switching sets based on switching set constraints
- When packaging to MPEG DASH, an adaptationset is created for each switchingset ingested
- The near constant fragment duration is used to generate segmenttemplate based presentation using either $Number$ or $Time$
- In case changes happen, the packager can update the manifest and embed inband events to trigger manifest updates in the fragments
- The DASH Packager encrypts media segments according to key information available. This key information is typically exchanged by protocol defined in Content Protection Interchange Format (CPIX) this allows configuration of the content keys, initialization vectors and embedding encryption information in the manifest
- The DASH packager signals subtitles in the manifest based on received CMAF streams and roles signalled in kind box
- In case a fragment is missing and SegmentTimeline is used, the packager may signal a discontinuity in the Manifest presentation description
- In case a low latency mode is used, the packager may make output available before the entire fragment is received in the chunked transfer encoding
- The packager may also have a proprietary API similar to the live source, for configuration of aspects like the segmentTimeBuffer, DVR window, encryption modes enabled etc.
- The packager uses a DASH or HLS ingest to push content to an origin server of content delivery network. Alternatively, it could also make content directly available as DASH or HLS as an origin server. In this case DASH/HLS ingest is avoided, and the packager also serves as the origin server.
- The packager converts the timed metadata track and uses it to convert to either MPD Events or inband events signalled in the manifest.
- The packager may also generate HLS or other streaming media presentations based on the input.
- In case the packager crashes or fails, it will restart itself and wait for the ingest source to perform the actions as detailed in the section on failover

The content delivery network (CDN) consumes a DASH/HLS ingest, or serves as a proxy for content delivered to a client. The CDN, in case it is consuming the POST based DASH/HLS ingest performs the following tasks

- it stores all posted content and makes them available for HTTP GET requests from locations corresponding to the paths signalled in the manifest
- it occasionally deletes content based on instructions from the ingest source, in this setup the packager

- in case low latency mode is used, content could be made available before the entire pieces of content are available
- It updates the manifest accordingly when a manifest update is received
- It serves as a cache proxy for HTTP get requests forwarded to the packager

In case the CDN serves as a proxy, it only forwards requests for content to the packager to receive the content,and caches relevant segments for a duration N until it expires.

The client receives DASH or HLS streams, and is not affected by the specification of this work. Nevertheless, it is expected that by using a common media application format, less caching and less overhead in the network will result in a better user experience. The client still needs to retrieve license and key information by steps defined outside of this specification. Information on how to retrieve this information will typically be signalled in the manifest prepared by the packager.

This example aims to illustrate how the specification defined in this document can be used to provide a live streaming presentation to clients, this example does not preclude other ways of using the specification and protocols defined in this document.

A second example can be seen in Diagram 12. It constitutes the reference workflow for chunked DASH CMAF under development by DASH-IF and DVB. In this workflow a contribution encoder produces an RTP mezzanine stream that is transmitted to FFMPEG, an open source encoder/packager running on a server. Alternatively, a file resource may be used. In this workflow FFMPEG functions as the ingest source. FFMPEG produces the ingest stream with different ABR encoded CMAF tracks. In addition, it also sends a manifest that complies with DASH-IF and DVB low latency CMAF specification and MPD updates. The CMAF tracks also contain respective timing information (prft etc.). In this case the ingest source implements interface 2 DASH ingest. But as in this case the DASH presentation uses CMAF, the media and track constraints of interface 1 are also satisfied. By also resending CMAF Headers in case of failures both interfaces may be satisfied.

The origin server is used to pass the streams to the client, and may in some cases also perform a re-encryption or re-packaging of the streaming presentation as needed by the client (in case encryption is needed for example). The target client is DASH.js and an end-to-end latency of maximum 3500 ms is targeted.

This example DASH reference workflow uses DASH Ingest that does not employ encryption and timed metadata and uses CMAF formatting. This exploits the synergies between the two interfaces defined in this document hence the ingest between FFMPEG and the origin server may implement both interfaces simultaneously.

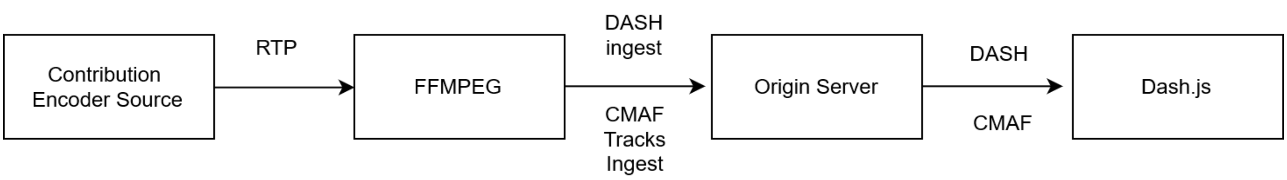
To receive the stream as a CMAF ingest for re-packaging at the origin the following steps can be applied.

1. Ignore the DASH Manifest
2. Ignore the segment names, only look at the relative path to identify the stream names
3. Ignore the HTTP Delete commands

The approaches for authentication and DNS resolution are similar for the two profiles/interfaces, as are the track formatting in case CMAF based media are used. This example does not use timed metadata. The ingest source may resend the CMAF header or init fragment in case of connection failures to conform to the CMAF ingest specification. The origin server can then be used to repackage or re-encrypt the streams.

To receive the stream as a DASH Ingest in this workflow, the steps described in DASH Ingest may be applied.

Diagram 12: DASH-IF Reference DASH-IF Live Chunked CMAF Production Workflow



10. IANA Considerations§

This memo includes no request to IANA.

11. Acknowledgements§

We thank the contributors to draft and the support from the following companies: Huawei, Akamai, BBC R&D, CenturyLink, Microsoft, Unified Streaming, Facebook, Hulu, Comcast, ITV, Qualcomm, Tencent, Samsung, MediaExcel, Harmonic, Sony, Arris, BitMovin, DSR and AWS Elemental.

12. References§

12.1. URL References§

fmp4git Unified Streaming github fmp4 ingest, "https://github.com/unifiedstreaming/fmp4-ingest".

MozillaTLS Mozilla Wikie Security/Server Side TLS https://wiki.mozilla.org/Security/Server_Side_TLS #Intermediate_compatibility_.28default.29 (last acessed 30th of March 2018)

MS-SSTR Smooth streaming protocol https://msdn.microsoft.com/en-us/library/ff469518.aspx last updated March 16 2018 (last acessed June 11 2018)

Conformance§

Conformance requirements are expressed with a combination of descriptive assertions and RFC 2119 terminology. The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in the normative parts of this document are to be interpreted as described in RFC 2119. However, for readability, these words do not appear in all uppercase letters in this specification.

All of the text of this specification is normative except sections explicitly marked as non-normative, examples, and notes. [RFC2119]

Examples in this specification are introduced with the words “for example” or are set apart from the normative text with `class="example"`, like this:

> ¶ EXAMPLE 1
> This is an example of an informative example.

Informative notes begin with the word “Note” and are set apart from the normative text with `class="note"`, like this:

> Note, this is an informative note.

Index§

Terms defined by this specification§

CMAF chunk

CMAF fragment

CMAF Header

CMAF Ingest

CMAF Media object

CMAF segment

CMAFstream

CMAF Track

Connection

DASH Ingest

eIng

fmp4git

ftyp

HLS Ingest

HTTP POST

Ingest source

Ingest Stream

Live encoder

Live stream event

Manifest objects

mdat

mdhd

Media fragment

Media objects

Media processing entity

mfra

moof

moov

MozillaTLS

MS-SSTR

nmhd

Objects

OTT

POST_URL

Publishing point

Receiving entity

RTP

Streaming presentation

Switching set

TCP

tfdt

References§

Normative References§

[DVB-DASH]
ETSI TS 103 285 Digital Video Broadcasting (DVB); MPEG-DASH Profile for Transport of ISO BMFF Based DVB Services over IP Based Networks. March 2018. URL: http://www.etsi.org/deliver/etsi_ts/103200_103299/103285/01.02.01_60/ts_103285v010201p.pdf

[ID3v2]
ID3 tag version 2.4.0 - Main Structure. URL: http://id3.org/id3v2.4.0-structure

[ISO-639-2]
ISO/TC 37/SC 2. Codes for the representation of names of languages -- Part 2: Alpha-3 code. 1998. International Standard. URL: https://www.iso.org/standard/4767.html

[ISOBMFF]
Information technology — Coding of audio-visual objects — Part 12: ISO Base Media File Format. December 2015. International Standard. URL: http://standards.iso.org/ittf/PubliclyAvailableStandards/c068960_ISO_IEC_14496-12_2015.zip

[MPEG2TS]
Information technology -- Generic coding of moving pictures and associated audio information: Systems ITU-T Rec. H.222.0 / ISO/IEC 13818-1:2013. URL: http://www.itu.int/rec/T-REC-H.222.0-201206-I

[MPEG4-30]
ISO/IEC 14496-30:2014 Preview Information technology -- Coding of audio-visual objects -- Part 30: Timed text and other visual overlays in ISO base media file format. March 2014. URL: https://www.iso.org/standard/63107.html

[MPEGCMAF]
ISO/IEC 23000-19:2018 Information technology -- Multimedia application format (MPEG-A) -- Part 19: Common media application format (CMAF) for segmented media. January 2018. URL: https://www.iso.org/standard/71975.html

[MPEGDASH]
ISO/IEC 23009-1:2014 Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats. URL: http://standards.iso.org/ittf/PubliclyAvailableStandards/c065274_ISO_IEC_23009-1_2014.zip

[MPEGHEVC]
ISO/IEC 23008-2:2013 Information technology -- High efficiency coding and media delivery in heterogeneous environments -- Part 2: High efficiency video coding. December 2013. URL: https://www.iso.org/standard/35424.html

[RFC1035]
P.V. Mockapetris. Domain names - implementation and specification. November 1987. Internet Standard. URL: https://tools.ietf.org/html/rfc1035

[RFC2119]
S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. March 1997. Best Current Practice. URL: https://tools.ietf.org/html/rfc2119

[RFC2626]
P. Nesser II. The Internet and the Millennium Problem (Year 2000). June 1999. Informational. URL: https://tools.ietf.org/html/rfc2626

[RFC2818]
E. Rescorla. HTTP Over TLS. May 2000. Informational. URL: https://tools.ietf.org/html/rfc2818

[RFC3986]
T. Berners-Lee; R. Fielding; L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. January 2005. Internet Standard. URL: https://tools.ietf.org/html/rfc3986

[RFC5234]
D. Crocker, Ed.; P. Overell. Augmented BNF for Syntax Specifications: ABNF. January 2008. Internet Standard. URL: https://tools.ietf.org/html/rfc5234

[RFC7235]

R. Fielding, Ed.; J. Reschke, Ed.. Hypertext Transfer Protocol (HTTP/1.1): Authentication. June 2014. Proposed Standard. URL: https://tools.ietf.org/html/rfc7235

[RFC7617]

J. Reschke. The 'Basic' HTTP Authentication Scheme. September 2015. Proposed Standard. URL: https://tools.ietf.org/html/rfc7617

[RFC793]

J. Postel. Transmission Control Protocol. September 1981. Internet Standard. URL: https://tools.ietf.org/html/rfc793

[RFC8216]

R. Pantos, Ed.; W. May. HTTP Live Streaming. August 2017. Informational. URL: https://tools.ietf.org/html/rfc8216

[SCTE35]

Digital Program Insertion Cueing Message for Cable. URL: https://www.scte.org/SCTEDocs/Standards/SCTE%2035%202019.pdf

↑

→