Guidelines for Implementation: DASH-IF Ingest

Living Document, 17 December 2018

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

Issue Tracking:
GitHub

Editors:
DASH IOP Ingest TF

Table of Contents

1 Specification: Live Media and Metadata Ingest
1.1 Abstract
1.2 Copyright Notice and Disclaimer

2 Introduction

3 Conventions and Terminology

4 Media Ingest Workflows

5 General Ingest Protocol Behavior

6 Profile 1: Fragmented MPEG-4 Ingest General Considerations

7 Profile 1: Fragmented MPEG-4 Ingest Protocol Behavior #
7.1 General Protocol Requirements
7.2 Requirements for formatting Media Tracks
7.3 Requirements for Timed Text Captions and Subtitle Streams
7.4 Requirements for Timed Metadata
7.5 Requirements for Media Processing Entity Failover
7.6 Requirements for Live Media Source Failover

8 Profile 2: DASH and HLS Ingest General Considerations

9 profile 2: DASH Ingest Protocol Behavior
9.1 General Protocol Requirements
9.2 Requirements for Formatting Media Tracks
9.3 Requirements for Timed Text Captions and Subtitle stream
9.4 Requirements for Timed Metadata
9.5 Requirements for Media Processing Entity Failover
9.6 Requirements for Live Media Source Failover

10 Security Considerations

11 IANA Considerations

12 Contributors

13 References
13.1 URL References

14 Author’s Address

Conformance

Index
Terms defined by this specification

References
Normative References
Informative References

1. Specification: Live Media and Metadata Ingest§

1.1. Abstract§

This draft presents a specification for ingesting media from a live source downstream. Two profiles are defined. The first profile is based on the fragmented MPEG-4 format. In this case the final media streaming presentation is generated downstream. The second profile enables ingest of media streaming presentations based on MPEG DASH and HLS directly using a common media format. Details on carriage of metadata markers, timed text, subtitles and encryption specific metadata are also included.

1.2. Copyright Notice and Disclaimer§

2. Introduction§

This document describes a specification for ingesting encoded media content from a ingest source such as a live video encoder downstream, for example towards distributed media processing entities. Examples of such entities include media packagers, publishing points, streaming origins and content delivery networks. The combination of ingest sources ingesting media and distributed media processing entities is important in practical video streaming deployments. In such deployments, interoperability between ingest sources and downstream processing entities can be challenging. This challenge comes from the fact that there are multiple levels of interoperability that need to be adressed and achieved.

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 disconnections and failures, procedures for repeatedly sending and receving 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, targetting several different application areas. To do so they provide many different profiles and options. Detailed operability is often achieved through other application standards such as those for the broadcast or storage. In addition, the codec and profile used, e.g. [MPEGHEVC] is an important interoperability point that itself also has different profiles.

A third level, is the way metadata is inserted in streams which can be a source of interoperability issues, especially for live content that needs such metadata to signal opportunities for signalling ad insertion, or other metadata like timed graphics. Examples of such metadata include [SCTE35] markers which are often found in broadcast streams and other metadata like ID3 tags [ID3v2].

Fourth, for live media handling the timeline of the presentation consistently is important. This includes correct sampling of media, avoiding timeline discontinuities and synchronizing timestamps attached by different ingest sources such as audio and video.

Fifth, in streaming workflows it is important to have support for failovers of both the live sources and the media processing entities. This is important to avoid interruptions of 24/7 live services such as Internet television where components can fail. In practical deployments, multiple ingest sources and media processing entities are used. This requires the multiple ingest sources and media processing to work together in a redundant workflow where some of the components might fail.

This document provides a specification for establishing this interoperability point. The approaches are based on known standardized technologies and have been tested and deployed in several streaming large scale streaming deployments. Two key workflows have been identified for which two different media ingest profiles will be detailed.

In first workflow, encoded media is ingested downstream for further processing of the media. Examples of such media processing could be any media transformation such as packaging, encrypting or transcoding the stream. Other operations could include watermarking, content insertion and generating streaming manifests based on [MPEGDASH] or HLS [RFC8216]. What is typical of these operations is that they actively inspect, or modify the media content and may generate new derived media content. In this workflow it is is important to convey mediadata and metadata that assists such active media processing operations. This is workflow type is adressed in the first profile fmp4 ingest.

In the second workflow, the encoded media is ingested into an entity that does none or very minimal inspection or modification of the media content. The main aim of such processing entities often lies in storage, caching and delivery of the media content. An example of such an entity is a Content Delivery Network (CDN) for delivering and caching Internet content. Content delivery networks are often designed for Internet content like web pages and might not be aware of media specific aspects. In fact, streaming protocols like MPEG DASH and HTTP Live Streaming have been developed with re-use of such a media agnostic Content Delivery Networks in mind. For ingesting encoded media into a content delivery network it is important to have the media presentation in a form that is very close or matching to the format that the clients need to playback the presentation, as changing or complementing the media presentation will be difficult. This second workflow is addressed in profile 2 DASH and HLS ingest.

```
Diagram 1: Example with media ingest in profile 1

============       ==============      ==============
||        || ingest||  Active   || HLS  || Content  ||  HLS
|| ingest ||====>>>||processing||===>>>|| Delivery ||==>>>Client
|| source ||       || entity   || DASH || Network  ||  DASH
============       ==============      ==============
```

```
Diagram 2: Example with media ingest in profile 2

============       ==============
||        || ingest|| Content  ||   HLS/DASH
|| ingest ||====>>>||Delivery  ||==>>>> Client
|| source ||       || Network  ||
============       ==============
```

Diagram 1 shows the workflow with a live media ingest from a ingest source towards an active media processing entity. In the example in diagram 1 the media processing entity prepares the final media presentation for the client that is delivered by the Content Delivery Network to a client.

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.

An example of a media ingest protocol is the ingest part of Microsoft Smooth Streaming protocol MS-SSTR. This protocol connects live encoders to the Microsoft Smooth Streaming server and to the Microsoft Azure cloud. This protocol has shown to be robust, flexible and easy to implement in live encoders. In addition it provided features for high availability and server side redundancy.

The first profile fmp4 ingest relating to workflow 1 advances over the smooth ingest procotol including lessons learned over the last ten years after the initial deployment of smooth streaming in 2009 and several advances on signalling of information such as timed metadata markers for content insertion. In addition, it incorporates the latest media formats and protocols, making it ready for current and next generation media codecs such as [MPEGHEVC] and protocols like MPEG DASH [MPEGDASH]. In addition, to address the sub profiling of exising media containers [MPEGCMAF] is referenced.

A second profile DASH and HLS ingest is included for ingest of media streaming presentations to entities were the media is not altered actively, and further media processing perhaps restricted to the manifests. A key idea of this part of the specification is to re-use the similarities of MPEG DASH [MPEGDASH] and HLS [RFC8216] protocols to enable a simultaneous ingest of media presentations of these two formats using common media segments such as based on [ISOBMFF] and [MPEGCMAF] formats. In addition, in this approach naming is important to enable direct processing and storage of the presentation.

Based on our experience we present these two separately. We made this decision as it will reduce a lot of overhead in the information that needs to be signalled compared to having both profiles combined into one, as was the case in a prior version of this draft.

We further motivate the specification in this document supporting using HTTP 1.1 [RFC7235] and [ISOBMFF] a bit more. We believe that Smooth streaming MS-SSTR and HLS [RFC8216] have shown that HTTP usage can survive the Internet ecosystem for media delivery. In addition, HTTP based ingest fits well with current HTTP based streaming protocols including [MPEGDASH]. In addition, there is good support for HTTP middleboxes and HTTP routing available making it easier to debug and trace errors. The HTTP POST provides a push based method for delivery for pusing the live content when available.

The binary media format for conveying the media is based on fragmented MPEG-4 as specified in [ISOBMFF] [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 fragmented MPEG-4 will make things easier and that the industry is already heading in this direction following recent specifications like [MPEGCMAF] and HLS [RFC8216].

Regarding the transports protocol, in future versions, alternative transport protocols could be considered advancing over HTTP 1.1 or TCP. We believe the proposed media format will provide the same benefits with other transports protocols. Our view is that for current and near future deployments using [RFC7235] is still a good approach.

The document is structured as follows, in section 3 we present the conventions and terminology used throughout this document. In section 4 we present use cases and workflows related to media ingest and the two profiles presented. Sections 5-9 will detail the protocol and the two different profiles.

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].

Ingest Stream: the stream of media produced by the live source transmitted to the media processing entity.

Live Stream Event: the total live stream for the ingest. (Live) encoder: entity performing live encoding and producing a high quality live stream, can serve as ingest source

Ingest source: a media source ingesting media content , typically a live encoder but not restricted to this.

Publishing point : entity used to publish the media content, consumes/receives the incoming media ingest stream

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

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

ftyp: the filetype and compatibility "ftyp" box as described in the ISOBMFF [ISOBMFF] that describes the "brand"

moov: the container box for all metadata "moov" described in the ISOBMFF base media file format [ISOBMFF]

moof: the movie fragment "moof" box as described in the ISOBMFF base media file format [ISOBMFF] that describes the metadata of a fragment of media.

mdat : the media data container "mdat" box contained in an ISOBMFF [ISOBMFF], this box contains the compressed media samples

mfra: the movie fragment random access "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" in the base media file format [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 media header box "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 box "elng" defined in [ISOBMFF] that can override the language information

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

HTTP POST : Command used in the Hyper Text Transfer Protocol for sending data from a source to a destination [[!RFC2626]

media fragment media fragment, combination of moof and mdat in ISOBMFF structure

fragmentedMP4stream : A fragmentedMP4stream can be defined using the IETF RFC 5234 ANB [RFC5234] as follows. **fragmentedMP4stream** = headerboxes fragments: headerboxes = ftyp moov fragments = X fragment fragment = Moof Mdat

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]

4. Media Ingest Workflows§

In this section we highlight two example workflows. Diagram 3 shows an example workflow of media ingest with fmp4 ingest in a streaming workflow. The live media is ingested into the media processing entity that performs 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 distributed media processing offloads many functionalities from the ingest source. As long as the stream originating from the media source contains sufficient metadata, the media processing entity can generate the media presentation for streaming to clients or other derived media presentations as needed by a client.

Diagram 4 shows an alternative example with ingest to a content delivery network, or perhaps another passive media entity such as a storage. In this case the live media source posts the segments and the manifests for the media presentation. In this case, still fragmented MPEG-4 segments can be used, but the ingest works slightly different.

Practice has shown that the ingest schemes can be quite different for the two configurations , and that combining them into a single protocol will result in overhead such as sending duplicate information in the manifest or ISOBMFF moov box, and increased signalling overhead for starting, closing and resetting the connection. Therefore, the two procedures for media ingest in such two common workflows are presented as separate profiles in the next two sections.

```
Diagram 3: Streaming workflow with fragmented MPEG-4 ingest in profile 1

============        ==============      ==============
|| live   ||ingest ||  Media    || HLS  || Content  ||  HLS
|| media  ||====>>>||processing||===>>>|| Delivery ||==>>> Client
|| source || fmp4  || entity    || DASH || Network  ||  DASH
============        ==============      ==============
```

```
Diagram 4:
Streaming workflow with DASH ingest in profile 2
============ingest ==============
|| live   || DASH  || Content  ||
|| media  ||====>>>||Delivery  ||==>>>> Client
|| source ||       || Network  ||
============        ==============
```

In Diagram 5 we highlight some of the key differences for practical consideration between the profiles. In profile 1 the encoder can be simple as the media processing entity can do many of the operations related to the delivery such as encryption or generating the streaming manifests. In addition the distribution of functionalities can make it easier to scale a deployment with many live media sources and media processing entities.

In some cases, an encoder has sufficient capabilities to prepare the final presentation for the client, in that case content can be ingested directly to a more passive media processing entity that provides a more pass through like functionality. In this case also manifests and other client specific information needs to be ingested. Besides these factors , chosing a workflow for a video streaming platform depends on many factors. The media ingest best practice covers these two types of workflows by two different profiles. The best choice for a specific platform depends on many of the use case specific requirements, circumstances and the available technologies.

```
  Diagram 5: Differences profile 1 and profile 2 for use cases
============================================================
|Profile   |    Ingest source      |   Media processing     |
|----------|----------------------|-----------------------|
|Profile 1 |limited overview      |encrypt, transcode,     |
|          | simple encoder       |package, watermark      |
|          | multiple sources     |content stitch, timed   |
|Profile 2 |Global overview       | cache, store, deliver  |
|          |encoder targets client|                        |
|          |only duplicate sources| manifest manipulation  |
|          |                      | storage & transmission |
============================================================
```

```
Diagram 6:
workflow with redundant Ingest sources and media processing entities

              fmp4   ==============
============ stream||  Media   ||
|| Ingest ||====>>>||Processing|| \\
|| source ||   //  ||  Entity  ||  \\
============  //   ==============   \\  ============
============ //                      \\ || load   ||
|| Ingest ||// redundant stream       >>||balancer|| ==>>> Client
|| source ||\\ stream                //  ============
============ \\     =============   //
============  \\   || Media     || //
|| Ingest ||====>>>||Processing ||//
|| source ||   //  || Entity    ||
============  //   ===============
============ //
|| Ingest ||// redundant stream
|| source ||
============
```

In Diagram 6 we highlight another aspect taken into consideration for large scale systems with many users. Often one would like to run multiple ingest sources, multiple downstream processing entities and make them available to the clients via a load balancer. This way requests can be balanced over multiple processing nodes. This approach is common when serving web pages, and this architecture also applies to video streaming platforms that also use [RFC7235]]. In Diagram 6 it is highlighted how one or more multiple live encoders can be sending data to one or more processing entities. In such a workflow it is important to handle the case when one source or media processing entity fails over. We call this support for failover. It is an important consideration in practical video streaming systems that need to run 24/7. Failovers must be handled robustly and seamlesslessly without causing service interruption. In both profiles we detail how this failover and redundancy support can be achieved.

5. General Ingest Protocol Behavior§

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

1. The live encoder or ingest source communicates to the publishing point /processing entity using the HTTP POST method as defined in the HTTP protocol [RFC7235]
2. The media ingest source SHOULD use HTTP over TLS [RFC2818] to connect to the media processing entity
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 dynamic 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 resillience to changes of the IP address in large scale deployments where the IP adress of the publishing point media processing nodes may change frequenty.
5. In case HTTPS [RFC2818] protocol is used, basic authentication HTTP AUTH [RFC7617] or better methods like TLS client certificates SHOULD be used
6. As compatibility profile for the TLS encryption we recommend the ingest SHOULD use the mozzilla intermediate compatibility profile which is supported in many available implementations MozillaTLS.
7. The encoder or ingest source SHOULD terminate the HTTP POST request if data is not being sent at a rate commensurate with the MP4 segment duration. An HTTP POST request that does not send data can prevent publishing points or media processing entities from quickly disconnecting from the live encoder or media ingest source in the event of a service update. For this reason, the HTTP POST for sparse data such as sparse tracks SHOULD be short-lived, terminating as soon as the sparse fragment is sent.
8. The POST request uses a POST_URL to the basepath of the publishing point at the media processing entity and MAY use a relative path for different streams and segments.

6. Profile 1: Fragmented MPEG-4 Ingest General Considerations§

The first profile assumes ingest to an active media processing entity, 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 using standardized media container formats based on [ISOBMFF] [MPEGCMAF]. In addition this allows extension to 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 7. Discussions on the early development have been documented fmp4git.

```
 Diagram 7: fragmented MPEG-4 ingest with multiple ingest sources

============ fmp4  ==============
||        || video ||          ||
|| ingest ||====>>>||          ||
|| source ||       ||          ||
============       ||          ||
||        || fmp4  ||          ||
|| ingest ||====>>>||  Active  ||      ==============
|| source || audio ||   Media  || HLS  || Content  ||  HLS
============       || procesing||===>>>|| Delivery ||==>>> Client
||        || fmp4  ||  entity  || DASH || Network  ||  DASH
||ingest  ||====>>>||          ||       =============
|| source || text  ||          ||
============       ||          ||
||        || fmp4  ||          ||
||ingest  || meta  ||          ||
|| source ||  data ||          ||
||        ||====>>>||          ||
============       ==============
```

In diagrams 8-10 we detail some of the concepts and structures. Diagram 8 shows the data format structure of fragmented MPEG-4 [ISOBMFF] and [MPEGCMAF]. In this format media meta data (playback time, sample duration) and sample data (encoded samples) are interleaved. the 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 header of the stream, sometimes referred as a [MPEGC MAF] header. The styp box can be used to signal the type of segment. The combination of styp moof mdat can be referred as a segment, the combination of ftyp and moof can be referred to as an init segment or a CMAF header.

```
Diagram 8: fragmented mp4 stream:

=========================================================
||ftyp||moov||styp||moof||mdat||styp||moof||mdat|| .....=
=========================================================
```

Diagram 9 illustrates the synchronisation model, that is in many ways similar, but simplified, from the synchronisation model propose in [MPEGCMAF]. Different bit-rate tracks and or media streams are conveyed in separate fragmented mp4 streams. by having the boundaries to the segments time alligned for tracks comprising the same 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 separated fragmented mp4 stream send over a separate connection, possibly from a different Ingest source. for more information on the synchronisation model we refer to section 6 of [MPEGCMAF].

In diagram 10 another advantage of this synchronisation model is illustrated, the concept of late binding. In the case of late binding a new stream becomes available. By using the segment boundaries and a common timeline it can be received by the [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.

```
Diagram 9: fmp4 stream synchronisation:

=========================================================
||ftyp||moov||styp||moof||mdat||styp||moof||mdat|| .....=
=========================================================
||ftyp||moov||styp||moof||mdat||styp||moof||mdat|| .....=
=========================================================
||ftyp||moov||styp||moof||mdat||styp||moof||mdat|| .....=
=========================================================
```

```
Diagram 10: fmp4 late binding:

===================================================
||ftyp||moov||styp||moof||mdat||moof||mdat|| .....=
===================================================
===================================================
||ftyp||moov||styp||moof||mdat||moof||mdat|| .....=
===================================================
```

Diagram 11 shows the flow of the media ingest. It starts with a DNS resolution (if needed) and an authentication step (Authy, TLS certificate) 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 segment). In case this is succesfull this is followed by the rest of the segments in the fragmented MPEG-4 stream. In the end of the session, for tear down the source can send an empty mfra box to close the connection.

```
Diagram 11: fmp4 ingest flow
||===============================================================||
||=====================           ============================  ||
||| live ingest source |            |  Media processing entity | ||
||=====================           ============================  ||
||         || <<------  DNS Resolve    -------->> ||            ||
||         || <<------  Authenticate   -------->> ||            ||
||         || <<------POST fmp4stream  -------->> ||            ||
||================== Moov + ftyp Sending  ======================||
||         || <<------ 404 Bad Request -----------||            ||
||         || <<------ 202 OK --------------------||            ||
||         || <<------ 403 Forbidden -------------||            ||
||         || <<------ 404 Bad Request            ||            ||
||         || <<------ 400 Forbidden -------------||            ||
||         ||          Unsupported Media Type     ||            ||
||         || <<------ 415 Forbidden -------------||            ||
||================== Moov + ftyp Sending  ======================||
||         || <<------ 202 OK --------------------||            ||
||============= fragmented MP4 Sending =========================||
||         || <<------ 404 Bad Request -----------||            ||
||============= mfra box Sending (close) =======================||
||         || <<------ 200 OK --------------------||            ||
||=====================           ============================  ||
||| live ingest source |            |  Media processing entity | ||
||=====================           ============================  ||
||         ||                                     ||            ||
||===============================================================||
```

7. Profile 1: Fragmented MPEG-4 Ingest Protocol Behavior #§

This section describes the protocol behavior specific to profile 1: fragmented MPEG-4 ingest. Operation of this profile MUST also adhere to general requirements in secion 4.

7.1. General Protocol Requirements§

1. The live encoder or ingest source SHOULD start by sending an HTTP POST request with the init segment by using the POSTURL This can help the live encoder or media ingest source to quickly detect whether the live ingest publishing point is valid, and if there are any authentication or other conditions required.
2. The live encoder or ingest source MUST initiate a media ingest connection by POSTING the CMAF header boxes "ftyp" and "moov" after step 1
3. The encoder or ingest source SHOULD use chunked transfer encoding option of the HTTP POST command [RFC2626] as it might be difficult to predict the entire content length of the segment. This can also be used for example to support use cases that require low latency.
4. If the HTTP POST request terminates or times out with a TCP error prior to the end of the stream, the encoder MUST issue a new connection, and follow the preceding requirements. Additionally, the encoder MAY resend the previous segment that was already sent again.
5. The live encoder or ingest source MUST handle any error or failed authentication responses received from the media processing, by issueing a new connection and following the preceding requirements inlcluding retransmitting the ftyp and moov boxes of the CMAF track header.
6. In case the live stream event is over the ingest source should signal the stop by transmitting an empty [=mfra] box towards the publishing point/processing entity.
7. The Ingest source SHOULD use a separate TCP connection for ingest of each different track

8. The Ingest source MAY use a separate relative path in the POST_URL for ingest of each different track by appending a relative path to the POST_URL
9. The fragment decode timestamps basemediadecodetime of fragments in the fragmentedMP4stream and the indexes base_media_decode_ time SHOULD arrive in increasing order for each of the different tracks/streams that are ingested.
10. The fragment sequence numbers of fragments in the fragmentedMP4stream signalled in the tfhd SHOULD arrive in increasing order for each of the different tracks/streams that are ingested.
11. Stream names MAY be signalled by adding the relative path Streams(stream_name) to the POST_URL
12. The average and maximum bitrate of each track SHOULD be signalled in the btrt box in the sampleentry of the CMAF header

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]
2. The trackFragmentDecodeTime box tfdt box MUST be present for each segment 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 MPEG-4 fragment durations SHOULD be between approximately 1 and 6 seconds.
5. The segments formatted as fragmented MP4 stream 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 considered.
6. The language of the stream SHOULD be signalled in the mdhd box or elng boxes in the init segment and/or moof headers (mdhd).
7. Segments posted towards the media procesing entity SHOULD contain the bitrate "btrt" box specifying the target average and maximum bitrate of the segments in the sample entry container in the init segment/CMAF header
8. The media track MAY use the notion of a CMAF chunk [MPEGCMAF] which is a moof mdat structure that may not be an IDR or switching point and is not targetted as an independently adressable media fragment
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.
10. Alternatively videotracks MAY use profiles like avc3 or hev1 that signal the parameter sets (PPS, SPS, VPS) in band in the media samples in the mdat box.

> 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. Alternatively signalling may be defined for the fragment, segment chunk structure.

7.3. Requirements for Timed Text Captions and Subtitle Streams§

The media ingest follows the following 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 the text in [MPEGCMAF] references prevails the text below when different.

1. The track will 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 fragmented MPEG-4 [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 2-5 must be present in the init segment (ftyp + moov) 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 init segment 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 btrt SHOULD be used to signal the average and maximum bitrate in the sample entry box

7.4. Requirements for Timed Metadata§

This section discusses the specific formatting requirements for ingest of timed metadata related to events and markers for ad insertion or other timed metadata An example of these are opportunities for dynamic live ad insertion 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]. For example, DASH Event messages contain a schemeIdUri that defines the payload of the message.

Table 1 provides some example urn schemes Table 2 illustrates an example of a SCTE-35 marker stored in a DASH emsg. The presented approach allows ingest of timed metadata from different sources, possibly on different locations by embedding them in sparse metadata tracks.

Example messages include e-msg [MPEGDASH], [DVB-DASH], [SCTE35] , [ID3v2]

```
Table 1 Example of DASH emsg schemes  URI
| Scheme URI                 | Reference                     |
| -------------------------:|:-----------------------------:|
| urn:mpeg:dash:event:2012   | DASH, 5.10.4                  |
| 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      |
```

```
Table 2 example of a SCTE-35 marker embedded in a DASH emsg
| Tag                    |          Value                    |
|-----------------------:|:---------------------------------:|
| scheme_uri_id          | urn:scte:scte35:2013:bin          |
| Value                  | the value of the SCTE 35 PID      |
| Timescale              | positive number                   |
| presentation_time_delta | non-negative number, splice time  |
|                        | relative  to tfdt                 |
| event_duration         | duration of event  "0xFFFFFFFF"   |
|                        |  indicates unknown duration       |
| Id                     | unique identifier for message     |
| message_data           | splice info section including CRC |
```

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

1. Create the metadata stream as a fragmentedMP4stream that conveys the metadata , the media handler (hdlr) is "meta", the track handler box is a null media header box nmhd.
2. The metadata stream applies to the media streams in the presentation ingested to active publishing point at the media processing entity
3. The URIMetaSampleEntry entry contains, 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]). For example, the URIBox could contain for ID3 tags [ID3v2] the URL http://www.id3.org or or urn:scte:scte35:2013a:bin for scte 35 markers [SCTE35]}
4. The timescale of the metadata should match the value specified in the media header box "mdhd" of the metadata track.
5. The Arrival time is signalled in the "tfdt" box of the track fragment as the basemediadecode time, this time when the metadata will be received
6. The application time can be signalled as a difference to the arrival time by having an empty sample with duration delta
7. The duration of the sample signalled in the trun box SHOULD correspond to the duration of the metadata
8. Empty samples, and fragments with empty samples SHOULD be used to fill the timeline
9. 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 in the metadata stream.
10. The metadata segment becomes available to the publishing point/ media processing entity when the corresponding track fragment from the media that has an equal or larger timestamp compared to the arrival time signalled in the tfdt basemediadecodetime. For example, if the sparse sample has a timestamp of t=1000, it is expected that after the publishing point/processing entity sees "video" (assuming the parent track name is "video") fragment timestamp 1000 or beyond, it can retrieve the sparse fragment from the binary payload.
11. The payload is conveyed in the mdat box as sample information. This enables muxing of the metadata tracks. For example XML metadata can for example be coded as base64 as common for [SCTE35] metadata messages
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.

7.5. 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 live encoder side, highly reliable live streaming setups can be build. In this section, we discuss requirements for failover scenarios.

The following steps are required for a live encoder or media ingest source to deal with a failing media processing entity.

1. Use a timeout for establishing the TCP connection. If an attempt to establish the connection takes longer abort the operation and try again.
2. Resend track segments for which a connection was terminated early
3. We recommend that the encoder or ingest source does 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 segment to be ingested. c. The new HTTP POST MUST include stream headers (ftyp, and moov boxes) identical to the stream headers in the initial POST request for fragmented media ingest.
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 manifest or segment 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
9. In case the media processing entity cannot proces a segment posted due to missing or incorrect init segment, an HTTP 412 unfulfilled condition SHOULD be returned
10. In case a media source receives an HTTP 412 response, it SHOULD resend ftyp and moov boxes

7.6. Requirements for Live Media Source Failover§

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

1. A ingest source instance SHOULD be instantiated to continue streaming
2. The ingest source MUST use the same URL for HTTP POST requests as the failed instance.
3. The new encoder or media ingest source POST request MUST include the same cmaf header box moov and ftyp as the failed instance
4. The new encoder or media ingest source MUST be properly synced with all other running encoders 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 start running at an appropriate segment boundaries.
5. The new stream MUST be semantically equivalent with the previous stream, and interchangeable at the header and media fragment levels.
6. The new encoder or media ingest source SHOULD try to minimize data loss. The basemediadecodetime tdft of media fragments SHOULD increase from the point where the encoder last stopped. The basemediadecodetime in the tdft box SHOULD increase in a continuous manner, but it is permissible to introduce a discontinuity, if necessary. Media processing entities or publishing points 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. Profile 2: DASH and HLS Ingest General Considerations§

Profile 2 is designed to ingest media into entities that only provide pass through functionality. In this case the media ingest source also provides the manifest based on MPEG DASH [MPEGDASH] or HTTP Live Streaming [RFC8216].

The key idea here is to reuse the fragmented MPEG-4 ingest to enable simulatneous ingest of DASH [MPEGDASH] and HLS [RFC8216] based on the fragmented MPEG-4 files using commonalities as described in [MPEGCMAF] which is a format based on fragmented MPEG-4 that can be used in both DASH and HLS presentations.

The flow of operation in profile 2 is shown in Diagram 12. In this case the Ingest source (media source) sends a manifest first. Based on this manifest the media processing entity can setup reception paths for the ingest url http://hostname/presentationpath In the next step segments are send in individual post requests using URLS corresponding to relative paths and segment names in the manifest. e.g. http://hostname/presentationpath/relative_path/segment1.cmf

This profile re-uses as much functionality as possible from profile 1 as the manifest can be seen as a complementary addition to the fragmented MPEG-4 stream. A difference lies in the way the connection is setup and the way data is transmitted, which can use relative URL paths for the segments based on the paths in the manifest. For the rest, it largely uses the same fragmented MPEG-4 layer based on [ISOBMFF] and [MPEGCMAF].

```
Diagram 12
||===============================================================||
||=====================           ============================  ||
||| live media source |           |  Media processing entity |  ||
||=====================           ============================  ||
||        ||                                     ||             ||
||===============Initial Manifest Sending=======================||
||        ||                                     ||             ||
||        ||-- POST /prefix/media.mpd  -------->>||             ||
||        ||          Succes                     ||             ||
||        || <<------ 200 OK --------------------||             ||
||        ||      Permission denied              ||             ||
||        || <<------ 403 Forbidden -------------||             ||
||        ||             Bad Request             ||             ||
||        || <<------ 400 Forbidden -------------||             ||
||        ||         Unsupported Media Type      ||             ||
||        || <<-- 412 Unfulfilled Condition -----||             ||
||        ||         Unsupported Media Type      ||             ||
||        || <<------ 415 Unsupported Media -----||             ||
||        ||                                     ||             ||
||=================== Segment Sending ==========================||
||        ||-- POST /prefix/chunk.cmaf  ------->>||             ||
||        ||          Succes/Accepted            ||             ||
||        || <<------ 200 OK --------------------||             ||
||        ||          Succes/Accepted            ||             ||
||        || <<------ 202 OK --------------------||             ||
||        ||      Premission Denied              ||             ||
||        || <<------ 403 Forbidden -------------||             ||
||        ||             Bad Request             ||             ||
||        || <<------ 400 Forbidden -------------||             ||
||        ||         Unsupported Media Type      ||             ||
||        || <<------ 415 Forbidden -------------||             ||
||        ||         Unsupported Media Type      ||             ||
||        || <<-- 412 Unfulfilled Condition -----||             ||
||        ||                                     ||             ||
||        ||                                     ||             ||
||=====================           ============================  ||
||| live media source |           |  Media processing entity |  ||
||=====================           ============================  ||
||        ||                                     ||             ||
||===============================================================||
```

9. profile 2: DASH Ingest Protocol Behavior§

Operation of this profile MUST also adhere to general requirements in section 5.

9.1. General Protocol Requirements§

1. Ingest source MUST send a manifest [MPEGDASH] with the following the limitations/constraints. 1a. Only relative URL paths to be used for each segment 1b. Only unique paths are used for each new presentation 1c. In case the manifest contains these relative paths, these paths SHOULD be used in combination with the POST_URL to HTTP POST each of the different segments from the live encoder or ingest source to the processing entity.
2. The live encoder or ingest source MAY send updated versions of the manifest, this manifest cannot override current settings and relative paths or break currently running and incoming POST requests. The updated manifest can only be slightly different from the one that was send previously, e.g. introduce new segments

available or event messages. The updated manifest SHOULD be send using a PUT request instead of a POST request.

3. Following media segment requests POST_URLs SHOULD be corresponding to the segments listed in the manifest as POST_URL + relative URLs.
4. The encoder or ingest source SHOULD use individual HTTP POST commands [RFC2626] for uploading media segments when available.
5. In case fixed length POST Commands are used, the live source entity MUST resend the segment to be posted decribed in the manifest entirely in case of responses HTTP 400, 404 412 or 415 together with the init segment consisting of "moov" and "ftyp" boxes
6. A persistent connection SHOULD be used for the different individual POST requests as defined in [RFC2626] enabling re-use of the TCP connection for multiple POST requests.
7. Each of fragment posted corresponds to a unique name+directory+timestamp
8. When the content lenght is known short formed POST SHOULD be used instead of a chunked transfer

9.2. Requirements for Formatting Media Tracks§

1. Media data tracks will be uploaded by individual segments, that use file extensions (.cmfv, .cfma etc.) to signal the type of content in the segment
2. Media specific information SHOULD be signalled in the manifest
3. Formatting described in manifest and media track MUST correspond consistently

9.3. Requirements for Timed Text Captions and Subtitle stream§

1. Timed Text, caption and subtitle stream tracks MUST be formatted conforming to the same requirements as in 6.3
2. Timed Text captions and subtitle specific information SHOULD also be signalled in the manifest
3. Formatting described in manifest and media track MUST correspond consistently

9.4. Requirements for Timed Metadata§

1. timed metadata is signalled using the higher level representation (DASH/HLS)

9.5. Requirements for Media Processing Entity Failover§

1. To be defined, including response codes

9.6. Requirements for Live Media Source Failover§

1. To be defined

10. Security Considerations§

Security consideration are extremely important for media ingest. Retrieving media from a illicit source can cause inappropriate content to be broadcasted and possibly lead to failure of infrastructure. Basic security requirements have been covered in section 5. No security considerations except the ones mentioned in this part of the text are expelictly considered. Further security considerations will be updated once they have been investigated further based on review of this draft.

11. IANA Considerations§

This memo includes no request to IANA.

12. Contributors§

Will Law Akamai

James Gruessing BBC R&D

Kevin Moore Amazon AWS Elemental

Kevin Johns CenturyLink

John Deutscher Microsoft

Patrick Gendron Harmonic Inc.

Nikos Kyriopoulos MediaExcel

Rufael Mekuria Unified Streaming

Sam Geqiang Zhang Microsoft

Arjen Wagenaar Unified Streaming

Dirk Griffioen Unified Streaming

Matt Poole ITV

Alex Giladi Comcast

13. References§

13.1. 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)

14. Author's Address§

Rufael Mekuria (editor) Unified Streaming Overtoom 60 1054HK

Phone: +31 (0)202338801 E-Mail: rufael@unified-streaming.com

Sam Geqiang Zhang Microsoft E-mail: Geqiang.Zhang@microsoft.com

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§

- basemediadecodetime
- Connection
- elng
- fmp4git
- fragmentedMP4stream
- ftyp
- HTTP POST
- Ingest source
- Ingest Stream
- Live Stream Event
- mdat
- mdhd
- media fragment
- Media processing entity
- mfra
- moof
- moov
- MozillaTLS
- MS-SSTR
- nmhd
- POST_URL
- Publishing point
- TCP

tfdt

References§

Normative References§

[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

[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%202016.pdf

Informative 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

↑

→