Specification of live Media Ingest

Living Document, 31 January 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 draft presents the specification of Live Media Ingest. Two profiles are defined, the first profile, CMAF ingest is based on fragmented MPEG-4 (fmp4). The second profile is based on MPEG DASH and HLS. Details on ingest of metadata markers, timed text, subtitles data are also included in the specification.

1.2. Copyright Notice and Disclaimer§

2. Introduction§

This document presents the specification 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. 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 there are multiple levels of interoperability that need to be 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 disconnects and failures, providing 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 inter-operability is often achieved through other application standards such as those for the broadcast or storage. For inter-operable live media ingest, this document provides guidance on how to use [ISOBMFF]. In addition, the codec and profile used, e.g. [MPEGHEVC] are important inter-operability 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 such metadata to signal opportunities for 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 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.

Fifth, in streaming workflows it is important to have support for failovers of both the ingest 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 that multiple ingest sources and media processing entities work together in a redundant workflow where some of the components might fail.

This document provides the specification for establishing these inter-operability points. The approaches are based on known standardized technologies that have been tested and deployed in several large scale streaming deployments. Two key workflows have been identified for which two different media ingest profiles will be detailed.

In a first workflow for profile 1, CMAF ingest, encoded media is ingested downstream for further processing of the media in the media processing entity. Examples of such media processing could be any media transformation such as packaging, encrypting or transcoding the streams. Other example operations 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 important to convey mediadata and metadata that assists such active media processing operations. This workflow type is addressed in the first profile referred to as CMAF ingest.

In the second workflow in profile 2, 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 supported in profile 2 DASH and HLS ingest. This specification provides a push based ingest for these protocols.

```
Diagram 1: Example with media ingest in profile 1 (fmp4/CMAF ingest)

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

```
Diagram 2: Example with media ingest in profile 2 (HLS or DASH ingest)

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

Diagram 1 shows the workflow with a live media ingest from an ingest source towards a 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/ingest sources 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 CMAF ingest 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 metadata and timed text. 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 support the sub profiling of exising media containers CMAF [MPEGCMAF] is referenced.

A second profile referenced as DASH and HLS ingest is included for ingest of media streaming presentations to entities were the media is not altered actively. 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 fragments such as based on [ISOBMFF] and [MPEGCMAF] formats. In this profile naming is important to enable direct processing and storage of the presentation.

We present these two profiles 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 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 format 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 format 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 main goal of this specification is to define the inter-operability 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]

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].

fmp4 Ingest: Ingest mode defined in this specification for ingesting fmp4 and/or CMAF content

CMAF Ingest: Ingest mode defined in this specification for ingesting CMAF content

DASH Ingest: Ingest mode defined in this specification for ingesting MPEG DASH content directly.

HLS Ingest: Ingest mode defined for ingesting HLS content directly

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 production of 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, it could be a storage source aswell.

publishing point : Entry point used to receive the media content, consumes/receives the incoming media ingest stream at the media processing entity

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

media objects objects ingested that representing media, and or timed text, or other non manifest objects

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

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

receiving entity: Entity used to process the media content, receives/consumes a media [=ingest stream]. May only provide a pass through functionality.

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

ftyp: the filetype and compatibility FileTypeBox "ftyp" box as described in the ISOBMFF [ISOBMFF]

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

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

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

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" 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 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, often used for metadata tracks

HTTP POST : Command used in the Hyper Text Transfer Protocol for sending data from a source to a destination [[!RFC7235]

media fragment Media fragment, combination of moof and mdat in ISOBMFF structure (MovieFragmentBox and mediaDataBox)

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]

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

Application time : The time a metadata event is applied to a stream (if applicable)

4. Media Ingest Workflows and Profiles§

In this section we highlight two example workflows corresponding to the two different profiles. Diagram 3 shows an example workflow of media ingest with fmp4 Ingest or CMAF 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 cloud storage. In this case the ingest source uses HTTP post to push the fragments and the manifests or other media objects composing the streaming presentation.

In this case, still MPEG-4 media fragments can be used, but the ingest works slightly different. The ingest can be based on DASH Ingest or HLS Ingest defined in profile 2 and includes sending the manifest. In this case the manifest and fragments are send with using individual fixed length HTTP POST commands to paths that correspond to paths defined in the manifest.

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 these two common workflows are presented as separately in different profiles.

```
Diagram 3: Streaming workflow with fragmented MPEG-4 ingest in profile 1 (CMAF/fmp4 ingest)

============       ==============      ==============
|| 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 DASH/HLS ingest

============  fmp4   ==============
|| live   ||manifest|| Content  ||
|| media  ||====>>> ||Delivery  ||==>>>> Client
|| source ||        || Network  ||
============        ==============
```

Diagram 5 highlights some of the key differences and practical considerations of the profiles. In profile 1 (CMAF or fmp4 ingest) the ingest source 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 pass through like functionality. In this case also manifests and other client specific information needs to be ingested. Besides these factors , choosing a workflow for a video streaming platform depends on many other factors. This specification does not provide guidance on what workflow is best to use in which cases. Yet, the live ingest specification covers the two types of workflows by two different profiles fmp4/CMAF ingest and DASH/HLS ingest. 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,     |
| CMAF     | simple encoder       |package, watermark      |
| fmp4     | multiple sources     |content stitch, timed   |
|Profile 2 |   Global overview    | cache, store, deliver  |
| DASH     |encoder targets client|                        |
| HLS      |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 ||
============
```

Diagram 6 highlights another aspect that was taken into consideration for large scale systems with many users. Often one would like to run multiple ingest sources, multiple media 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. In Diagram 6 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 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 such as Internet Television. Failovers must be handled robustly and seamlesslessly without causing service interruption. This specification details how this failover and redundancy support can be achieved. In addition, some recommendations for redundant ingest sources and media processing entities are provided.

5. General Ingest Protocol Behavior§

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

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 resillience to changes of the IP address in large scale deployments where the IP address of the media processing entities may change frequenty.
5. In case HTTPS [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 sparse 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. Profile 1: CMAF Ingest General Considerations§

The fmp4 or CMAF ingest 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 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 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 ||          ||
||        ||====>>>||          ||
============       ==============
```

Diagrams 8-10 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 such as 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 [MPEGCMAF] header. The styp box can be used to signal the type of fragment. The combination of styp moof mdat can be referred as a CMAF fragment, the combination of ftyp and moov can be referred to as an init fragment 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 based on the synchronisation model proposed in [MPEGCMAF]. Different bit-rate tracks and/or media streams are conveyed in separate fragmented mp4 streams. By having the boundaries to the fragments 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]. For synchronization of tracks coming from different encoders, sample time accuracy is required. i.e. the same samples need to be mapped to the sameple 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 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 or is adopted 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.

```
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|| .....=
===================================================
===================================================  << ==  different available track
||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 fragment) or cmaf header. In case this is succesfull this is followed by the rest of the fragments in the fragmented MPEG-4 stream. 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.

```
Diagram 11: fmp4 ingest flow
||==============================================================||
||=====================           ============================  ||
|||     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: CMAF Ingest Protocol Behavior§

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

7.1. General Protocol Requirements§

1. The ingest source SHALL start by sending an HTTP POST request with the init fragment "ftyp" and "moov" 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 or boxes 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 prior to the end of the stream, 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 occured.
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 track

8. The Ingest source MAY use a separate relative path in the POST_URL for ingest of each different track by appending it to the POST_URL
9. The fragment decode timestamps basemediadecodetime in tdft of fragments in the fragmentedMP4stream SHOULD arrive in increasing order for each of the different tracks/streams that are ingested.
10. The fragment sequence numbers seq_num of fragments in the fragmentedMP4stream signalled in the tfhd SHOULD arrive in increasing order for each of the different tracks/streams that are ingested. Using both timestamp in 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

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 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 fragment, cmaf header and/or moof headers (mdhd).
7. Fragments posted towards the media procesing entity 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 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 addressable 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 in the init fragment (CMAF Header).
10. Alternatively videotracks MAY use profiles like avc3 or hev1 that signal the parameter sets (PPS, SPS, VPS) in in the media samples in the mdat box.
11. In case the language of track changes a new init fragment with update mdhd and or elng SHOULD be send.

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. For example the styp box can be used to signal if the following structure is a CMAF fragment, chunk, or segment.

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 except for the notion of 9 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 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 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 init fragment 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 track changes a new init fragment with updated mdhd and/or elng SHOULD be send from the ingest source to the media processing entity.

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. In this approach metadata is not interleaved with the media as for example the case in emsg boxes in [!MPEGCMAF]

Example metadata messages include e-msg as used in [MPEGDASH], [DVB-DASH], or alternatively [SCTE35] or [ID3v2] messages could be conveyed in a sparse metadata track.

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,similar to track
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. Metadata SHALL be conveyed in a fragmented mp4 stream, where 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 ingested to a publishing point entry at the media processing entity
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]). 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 is the time when the metadata will be first detected.
6. 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 the media presentation time of the sample containing the event/metadata
7. The duration of the sample signalled in the trun box containing the metadata SHOULD correspond to the duration of the metadata if the metadata is valid for a duration of time (if applicable)
8. Empty samples, and fragments with empty samples SHOULD be used to fill the timeline to avoid timeline gaps
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.
10. The metadata fragment becomes available to the 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 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.

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 emsg 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.

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 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.

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 a ingest source receives an HTTP 412 response, it SHALL resend ftyp and moov boxes

7.6. 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 start running at an appropriate fragment 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 instance of 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 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 a streaming presentation composed of manifest objects and media objects in 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.

Profile 2 exists independently of Profile 1. The requirements below encapsulate all needed functionality to support Profile 2. The requirements listed for Profile 1 in section #general_Protocol_Requirements_p1 do not apply to Profile 2. General shared requirements are covered in section #general.

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). This capability is further illustrated in the Examples section TBD

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 differentation 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:

```
| File extension| Mime-type|
| --- | --- |
| .m3u8| application/x-mpegURL or vnd.apple.mpegURL |
| .mpd | video/MP2T |
| .ts | video/MP2T |
|.cmfv | video/mp4 |
|.cmfa | audio/mp4 |
|.cmft | application/mp4 |
|.m4v | video/mp4 |
|.mp4 | video/mp4 or application/mp4 |
|.m4a | audio/mp4 |
|.m4s | video/iso.segment |
|.init | video/mp4 |
|.header | video/mp4 |
|.key | to be defined |
```

8.1.4. DNS lookups§

1. The ingest source MUST perform a fresh DNS lookup of the receiving entity hostname prior to ingesting any manifest or media object at the start live stream event
2. The publishing entity MUST honor DNS Time To live values when re-connecting, for any reason, to the receiving entity.
3. For services in which ingest sources are dynamically allocated based upon DNS resolution, it is recommended that short TTL values are chosen in order to allow ingest sources to fail over to new ingest servers if warranted under a failure scenario.

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 DVR 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 an discontinuity marker appropriate for the protocol format at hand.
2. HTTP 403 or 400 errors a 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 manifests 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 XX appropriate for the mime-type of the content they are carrying.

8.3.2. Relative paths§

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

9. Security Considerations§

Security considerations are extremely important for media ingest. Retrieving media from an 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.

10. IANA Considerations§

This memo includes no request to IANA.

11. Acknowledgements§

We thank some of the contributors for contribution to draft or discussions

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

Cenk Facebook

Thomas Stockhammer Qualcomm

Iraj Sodaghar, Tencent

Ali Begen Comcast/Ozyegin University

Paul Higgs, Huawei

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)

13. 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§

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

[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%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