Workgroup: Individual
Internet-Draft: draft-ietf-hesp-04
Published: 12 April 2023
Intended Status: Informational
Expires: 14 October 2023
Authors: P. Speelmans, Ed.
THEO Technologies

HESP - High Efficiency Streaming Protocol

Abstract

This document describes a protocol for delivering multimedia data, enabling ultra-low latency and fast channel change over HTTP networks. It specifies the data format of the files and the actions to be taken by the server (sender) and the clients (receivers) of the streams. It describes version 2 of this protocol.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 14 October 2023.

Table of Contents

1. Introduction

Viewers are more demanding than ever, calling for a streaming protocol that combines ultra-low latency, fast zapping and cost-effective scalability.

HESP is an HTTP-based streaming approach that works with standard contribution feeds from (live) productions, standard encoders, albeit specifically configured, a HESP compliant packager, regular CDNs and an HESP compliant player.

HESP offers sub-second latency, near real-time interactivity, fast startup and channel change times and cost-effective scalability up to millions of viewers.

The purpose of this document is to facilitate interoperability between HESP implementations by describing the media transmission protocol.

This document describes version 2 of the protocol.

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 [RFC2119].

2. Overview

This section contains an overview of the HESP protocol and its building blocks.

2.1. HESP components

HESP follows a regular approach to online video streaming. The content is ingested and transcoded in different qualities. Each quality requires two streams. The encoded streams are packaged by an HESP packager and made available via an origin server. An HESP player requests the stream using HTTP requests. A CDN can be used.

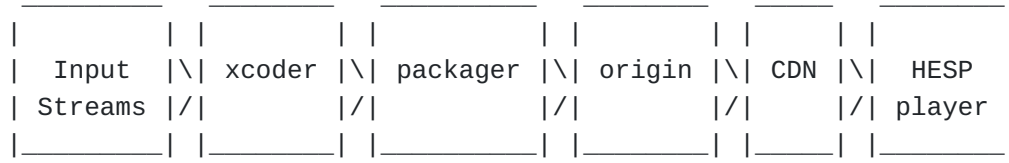


Figure 1: HESP chain from source to playback

2.2. Two complementary streams

HESP is based on using two streams for each track, the Initialization Stream and the Continuation Stream. The encoder MUST ensure that the corresponding media data of both streams are issued with synchronized presentation timestamps. The packager MUST ensure this data remains in sync.

The Initialization Stream consists of Initialization Packets. Initialization Packets MUST be individually addressable. An Initialization Packet MUST contain an independent media sample, the reference to the segment and the position in the segment where the Continuation Stream can start. Since the packet starts with an independent sample, playback can start with any Initialization Packet.

The Continuation Stream can start playback immediately after an Initialization Packet, allowing for very fast channel start and switch times. This mechanism puts referencing limitations to the Continuation Stream. When a reference is made from the Initialization Stream to a Continuation Stream, all data needed to render the sample with the subsequent Presentation timestamp MUST be available at the reference offset. In addition, the samples in the Initialization Stream and the samples in the Continuation Stream MUST be aligned. That is, the corresponding media samples of both streams MUST have the same PTS and MUST be made available at the same time. The Continuation Stream is addressed using byte-range requests. The Continuation Stream SHOULD be published in chunks in order to reduce end-to-end latency.

The player receives the Initialization Packet, initializes the decoder, puts the media data in the decoder buffer, and requests the subsequent media data from the Continuation Stream (using HTTP Range requests, starting by the range offset given by the Initialization Packet.)

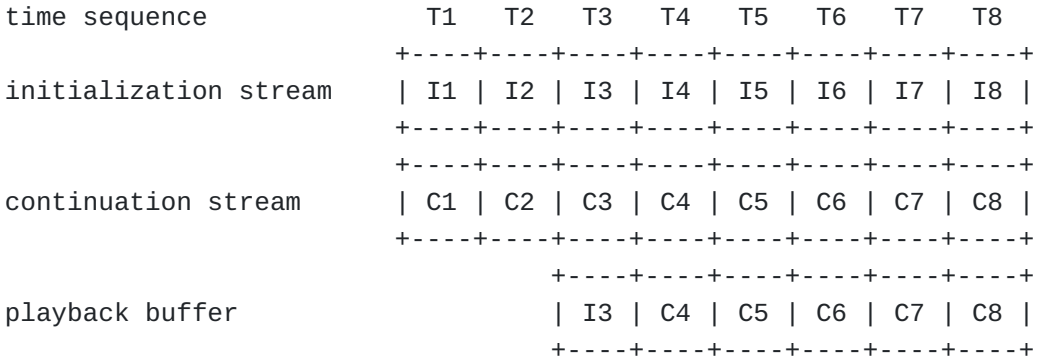


Figure 2: Start of an HESP stream

2.3. HESP object model

The object model follows the CMAF media object model [CMAF].

A Track is used to contain media samples (audio or video) or metadata. It consists of a Continuation Stream and (except for metadata Tracks) an Initialization Stream.

The Initialization Stream consists of Initialization Packets, where each such packet MUST contain a CMAF Header and possibly a CMAF Fragment. Each Initialization Packet MUST be individually addressable through Sequence Numbers. It is further explained in Section 4.

The Continuation Stream consists of Continuation Segments. These Segments MUST consist of CMAF Chunks that can be sent individually to clients. It is further explained in Section 5.

A Switching Set groups together Tracks that contain the same content but with different encoding parameters (e.g., different resolution or different bitrate). A client is able to seamlessly switch between Tracks of a Switching Set as a result. If multiple Switching Sets contain differing content but are aligned in their timings (e.g., multiple view perspectives on the same performance or different languages of the same audio), we can consider them Aligned Switching Sets.

A Selection Set groups together multiple Switching Sets of the same media type. HESP currently allows 3 Selection Sets: audio, video and metadata Selection Sets.

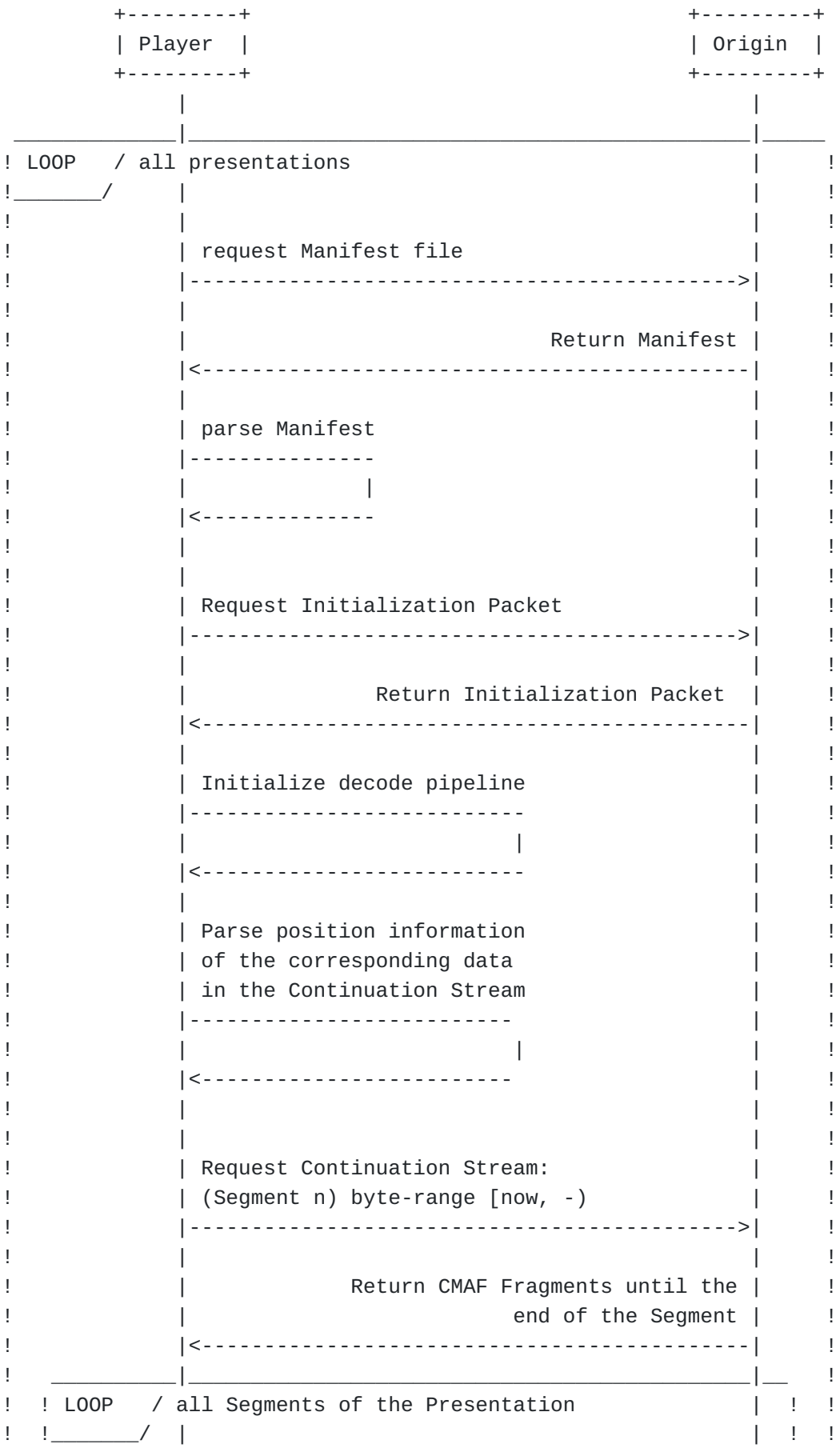
A Presentation contains the Selection Sets for a given period of time. Multiple Presentations together form a continuous timeline of media content, even though each Presentation might have different content, encodings or timestamps. It can be an advertisement, or a part of a show, or the first half of a game,... The Presentation is the lowest granularity inside a Manifest. All Tracks of a Presentation MUST have media data for the full duration of the Presentation.

The Manifest informs clients of the aforementioned data structure and must be retrieved before any media data. The format of the Manifest is detailed in Section 3.

A new Manifest is not needed to request a new Segment. Segment addressing can happen automatically within a Presentation for an efficient and continuous delivery of the Continuation Stream. A Manifest is only obligated to be updated before the start of a new Presentation. This gives the opportunity for low frequency Manifest updates, though this update rate can be freely configured.

Since Presentations can be of unknown duration, a different mechanism is used to signal a new Presentation to a client. To that extent, in-band metadata events are introduced in the Continuation Stream. Such an event can signal the client to retrieve a new Manifest, when information about an upcoming Presentation becomes available.

2.4. Reference flow


Player
Origin
LOOP / all presentations
request Manifest file
Return Manifest
parse Manifest
Request Initialization Packet
Return Initialization Packet
Initialize decode pipeline
Parse position information
of the corresponding data
in the Continuation Stream
Request Continuation Stream:
(Segment n) byte-range [now, -)
Return CMAF Fragments until the
end of the Segment
LOOP / all Segments of the Presentation

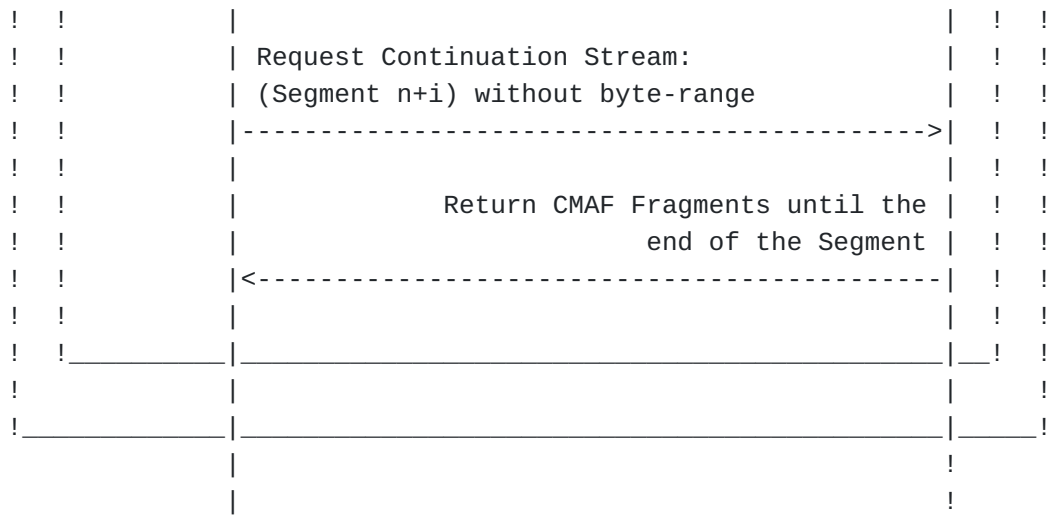

Request Continuation Stream:
(Segment n+i) without byte-range
Return CMAF Fragments until the
end of the Segment

Figure 3: HESP reference flow

Though the manifest file is shared, video, audio and metadata (subtitles) media data MUST be distributed separately. They each require separate requests from the client.

3. HESP Manifest

The first step for playback of an HESP stream is to fetch a Manifest. It contains information on the available Tracks and how to request content from every Track's Initialization and Continuation Streams.

3.1. Timestamps

3.1.1. Manifest Timestamp and Media Timestamp

A distinction is made between Manifest Timestamps and Media Timestamps in the definitions below. Media Timestamps MUST represent Presentation timestamps as they are given by the media data itself. Manifest Timestamps MUST define those same Media Timestamps, but with an additional offset such that the timestamps of all Tracks of a Presentation are aligned. This offset MUST be given by the Manifest for each Track or Switching Set.

For example, consider the first Segment of a Track with Media Timestamps starting at 0 seconds. This Track is part of the second Presentation of an HESP stream. The first Presentation has run for a significant amount of time and the second Presentation follows immediately after the first Presentation ends. In this case, the Manifest Timestamps of the second Presentation must succeed the Manifest Timestamps of the first Presentation. As a result, the Media Timestamps and Manifest Timestamps of the second Presentation now differ. An offset must be given by the Manifest in order to inform clients about this difference between timestamp types for each Track of the second Presentation.

An example of this distinction is given in Appendix A.1.2.

3.1.2. Sequence Numbers

Each Initialization Packet belonging to a Track is given a Sequence Number. This is a simple identifier used to retrieve specific Initialization Packets by the client. The Sequence Number is a positive integer that MUST be increased by 1 for each subsequent Initialization Packet of the same Track. The first Initialization Packet of all Tracks in all Switching Sets inside of a Presentation MUST contain the Manifest Timestamp for the start of that Presentation within the interval of the first presentation time of

that Initialization Packet and the sum of the presentation time and all durations of the frames in that Initialization Packet.

3.1.3. Calculating the Sequence Number of an Initialization Packet

To calculate the Sequence Number of a chosen Initialization Packet, the Manifest provides the client multiple values. Each active Presentation MUST contain both its current Manifest Timestamp and, for each Track, the Sequence Number of the first Initialization Packet and the (constant) frame rate.

It is given that:

- the frame rate MUST be constant for each Track.
- Initialization Packets MUST become available in real-time.
- Sequence Numbers MUST be increased by 1 per Initialization Packet.
- the Sequence Number and Manifest Timestamp of the first Initialization Packet for a Track are given by the Manifest.

As a result, it is possible to derive Sequence Numbers for arbitrary Initialization Packets. By taking the initial Sequence Number of a Track, its associated Manifest Timestamp and the frame rate, it is possible to calculate the number of frames between the initial Sequence Number and any later Sequence Number of an Initialization Packet. For example, for a Video Track the full calculation can be performed as follows: calculateSequenceNumber(timestamp) = floor((timestamp - presentation.timeBounds.startTime / presentation.timeBounds.scale) * videoTrack.frameRate) + videoTrack.startSequenceNumber

For example, suppose the start Sequence Number would be 34 at Manifest Timestamp 00:00:01.360 for a Track with a frame rate of 25 fps. To find the Sequence Number at Manifest Timestamp 00:00:04.120, the difference between both frames should be calculated. This means that the Sequence Number for the provided Manifest Timestamp is (4.120s - 1.360s) * 25fps + 34 = 103.

In case the encoder suffers from drift and is not synchronized with a wall clock, a packager SHOULD compensate for this drift by creating a new Presentation.

3.2. Manifest data types

The Manifest MUST be formatted as a JSON file. All data types used in the information below MUST satisfy the JavaScript Object Notation specification [RFC8259]. In regard to the definition of numbers within the JSON specification, their range and precision are implementation-dependent. It should be noted that for

interoperability, numbers SHOULD be representable using IEEE 754 double-precision numbers.

Some additional data types are introduced on top of the JSON specification to narrow the allowed values of some fields further.

3.2.1. Additional JSON data types

3.2.1.1. Integer

An integer (or int) is a subset of the number type defined in the JSON specification, limited to integer numbers. Concretely, following Section 6 of the JSON specification, a number MUST NOT have a fraction part to be considered an integer. Note that as mentioned above, integers SHOULD be representable using IEEE 754 double-precision numbers. As a result, integers SHOULD be within the [(-2**53)+1, (2**53)-1] range to make sure implementations agree on the exact numeric value.

3.2.1.2. Unsigned integer

An unsigned integer (or uint) is a subset of the integer type, only allowing for positive integer numbers. Concretely, following Section 6 of the JSON specification, an integer MUST NOT contain a minus to be considered an unsigned integer.

3.2.1.3. Enumeration

An enumeration defines a list of constant strings, where a valid value of this type MUST equal one of these strings. The type is written as Enum(x, y, z), where x, y and z are the constant strings allowed to be used.

3.2.1.4. DateTime

A DateTime MUST be formatted as a string in the format defined by ISO 8601 [ISO8601]:

```
YYYY-MM-DDThh:mm:ss.mmmTZD

where YYYY = four-digit year
      MM = two-digit month
      DD = two-digit day of the month
      hh = two digits of the hour (00 through 23)
      mm = two digits of a minute (00 through 59)
      ss = two digits of a second (00 through 59)
      mmm = three digits of a millisecond (000 through 999)
      TZD = time zone designator (Z or +hh:mm or -hh:mm)
```

3.2.2. ManifestType

This data type represents the root of the Manifest. The structure consists of general playback information and a collection of Presentations.

The table below gives the possible fields. The "Required?" column indicates if a field is REQUIRED (Y) or OPTIONAL (N).

Attribute	Type	Req?	Description
availabilityDuration	ScaledValue	Y	The amount of time (in seconds) that live content MUST be kept available to be retrieved by a client. It is used to define an interval [live - n, live], where live is the current live point and n is availabilityDuration, where content MUST be available for clients of a live stream. For VOD streams, this value MUST be ignored.
creationDate	DateTime	Y	The timestamp of the moment that this specific Manifest was created by the packager (in local packager time.)
fallbackPollRate	uint	Y	The number of seconds a player SHOULD wait to poll a new Manifest, if it hasn't requested any since retrieving the current Manifest.
manifestVersion	Enum("2.0.0")	Y	The version number of the Manifest.
presentations	Presentation Type[]	Y	A list of all Presentations that are currently known.
streamType	Enum("live", "vod")	Y	Indicates whether the stream is a live stream (live) which does not have a known ending, or a video on demand stream (vod) with a known ending and duration.
activePresentation	string	N	The identifier of the currently active

Attribute	Type	Req?	Description
currentTime	ScaledValue	N	Presentation. This value MUST be available if streamType equals live, but MUST be ignored if streamType equals vod. The most recent composition timestamp of any Track contained by this Manifest when this Manifest is generated. It SHOULD be specified in Manifest Time, not in Media Time (as that could have different offsets from Track to Track.) This value MUST be available if streamType equals live, but MUST be ignored if streamType equals vod.
contentBaseUrl	string	N	The base URL for all content requests relating to this Manifest. See Section 3.4 for more information.
timeSource	TimeSource	N	A reference to a time server with which the packager is synced. This value MUST be ignored if streamType equals vod.

Table 1

3.2.3. TimeSource

A TimeSource is used to sync the packager and player internal clocks to make requests for data only once it is available.

Attribute	Type	Required	Description
scheme	string	Y	A scheme id for the time source as defined as a valid @schemeIdURI in [DASH] section 5.8.4.11 Table 32.
value	string	Y	The information indicating where the time source can be found as indicated in [DASH] section 5.8.5.7.

Table 2

3.2.4. ScaledValue

In order to avoid rounding issues introduced through floating-point numbers, this structure defines two integers. The total value is calculated by dividing value over scale.

Attribute	Type	Required	Description
value	int	Y	The defined integer value.
scale	uint	N	If not defined, the scale SHALL equal 1.

Table 3

3.2.5. PresentationType

This is the type definition of a Presentation.

Attribute	Type	Req?	Description
id	string	Y	The unique identifier for this Presentation. It MUST be unique over all Presentations of this Manifest. It MUST NOT change over Manifest updates.
timeBounds	TimeBounds	Y	The time boundaries of this Presentation, in Manifest Time. The start time MUST be announced at least 2 seconds before the Presentation is active. The end time of an active Presentation MUST be available at least 2 seconds before the actual end of that Presentation. The startTime of the TimeBounds MUST be equal to the time corresponding to the startSequenceNumber of each Track.
audio	AudioSwitchingSetType[]	N	The audio Selection Set. It contains all audio Switching Sets of this Presentation. If not defined, this value

Attribute	Type	Req?	Description
			SHALL equal an empty list.
baseUrl	string	N	The base URL of this Presentation. It is part of the content base URLs for all Switching Sets belonging to this Presentation. See Section 3.4 for more information.
events	PresentationEventType[]	N	List of all currently available Presentation Events related to this Presentation. If not defined, this value SHALL equal an empty list.
metadata	MetadataSwitchingSetType[]	N	The metadata Selection Set. It contains all metadata Switching Sets of this Presentation. If not defined, this value SHALL equal an empty list.
video	VideoSwitchingSetType[]	N	The video Selection Set. It contains all video Switching Sets of this Presentation. If not defined, this value SHALL equal an empty list.

Table 4

3.2.6. TimeBounds

A TimeBounds structure denotes a time interval with a start and end time. A start or end time may be undefined. The desired behavior in such a situation depends on the specific usage of the structure.

These boundaries are inclusive at the start time and exclusive at the end time. I.e., if two TimeBounds need to be continuous, then the end time of the first one must equal the start time of the second.

Attribute	Type	Required	Description
startTime	uint	N	

Attribute	Type	Required	Description
			This value denotes the start time in seconds when divided by the timescale.
endTime	uint	N	This value denotes the end time in seconds when divided by the timescale.
scale	uint	N	If the timescale is not defined, it SHALL equal 1.

Table 5

3.2.7. AudioSwitchingSetType

This is the type definition of an audio Switching Set.

Attribute	Type	Req?	Description
id	string	Y	The unique identifier for this Switching Set. It MUST be unique within its Presentation.
language	string	Y	The language of all audio Tracks of this Switching Set. It MUST be specified here by its ISO 639-2 [ISO6392] code.
tracks	AudioTrackType[]	Y	The collection of all Tracks belonging to this Switching Set.
alignId	string	N	A unique identifier that SHOULD be set by all Switching Sets that are Aligned Switching Sets with each other.
baseUrl	string	N	The base URL of this Switching Set. It is part of the content base URLs for all Tracks belonging to this Switching Set. See Section 3.4 for more information.
channels	uint	N	The audio channel configuration of all audio Tracks belonging to this Switching Set. It is defined as the total number of audio

Attribute	Type	Req?	Description
			channels. This value is used for ABR selection by the player.
codecs	string	N	The definition of the codec(s) necessary to render the content of all Tracks belonging to this Switching Set. It MUST follow the ISO File Format Name Space as defined by [RFC6381]. If this is not defined, it MUST be defined on all Tracks of this Switching Set separately.
continuationPattern	string	N	The URL pattern used to request Continuation Segments belonging to this Switching Set. The pattern MUST include the {segmentId} string; see Section 3.4 for more information. If this is not defined, it MUST be defined on all Tracks of this Switching Set separately.
initializationPattern	string	N	The URL pattern used to request Initialization Packets belonging to this Switching Set. The pattern MUST include the {initId} string; see Section 3.4 for more information. If this is not defined, it MUST be defined on all Tracks of this Switching Set separately.
label	string	N	

Attribute	Type	Req?	Description
			A human-readable name for this Switching Set.
mediaTimeOffset	ScaledValue	N	The offset to be added to Manifest Timestamps to calculate Media Timestamps contained by Segments of this Switching Set. If neither the Track nor the Switching Set has this value set, it SHALL equal 0.
mimeType	string	N	The MIME type of all content belonging to this Switching Set. It MUST be a valid audio MIME type. If not set, the MIME type of all content of this Switching Set SHALL equal audio/mp4.
protection	SwitchingSet Protection	N	The information related to content protection for all Tracks belonging to this Switching Set.
sampleRate	uint	N	The sample rate (in Hz) of all audio Tracks belonging to this Switching Set. If this is not defined, it MUST be defined on all Tracks of this Switching Set separately.
samplesPerFrame	uint	N	The number of audio samples in one frame. If set, this MUST apply to each Track belonging to this Switching Set that does not define this value itself. If neither the Track nor the Switching Set has this value set, it SHALL equal 1024.

Table 6

3.2.8. SwitchingSetProtection

The following fields are defined on the SwitchingSetProtection structure:

Attribute	Type	Req?	Description
type	Enum("cenc", "cbcs")	Y	The protection scheme used to encrypt this Switching Set.
systems	SwitchingSetProtectionSystem[]	Y	Metadata about the DRM systems that can be used to playback this Switching Set. This list MUST contain at least one entry.

Table 7

More information on content protection can be found in Section 7.

3.2.9. SwitchingSetProtectionSystem

The following fields are defined on the SwitchingSetProtectionSystem structure:

Attribute	Type	Required	Description
pssh	string	N	A Base 64 encoded ProtectionSystemSpecificHeaderBox. If it is not defined in the Manifest, then such a box MUST be available in the Initialization Stream of each Track belonging to this Switching Set.
schemeId	string	Y	A UUID [UUID] that uniquely identifies the content protection system. It should match the System ID included in the pssh box of this protection system.

Table 8

Additional attributes, such as license acquisition URLs, authorization-related URLs, specific initialization data (a default key ID) or others, MAY be defined for this structure, depending on the scheme identifier. Below, additional attributes that MUST be included are given for a subset of System IDs. If a System ID is not included here, then additional attributes MAY still be defined by

the author of the content protection system and is considered out of scope of this document.

System ID	Attributes	Reference
94ce86fb-07ff-4f43-adb8-93d2fa968ca2	FairPlayAttributes	Apple FairPlay Streaming

Table 9

3.2.9.1. FairPlayAttributes

The following fields MUST be parsed from the SwitchingSetProtectionSystem structure when schemeId equals 94ce86fb-07ff-4f43-adb8-93d2fa968ca2. More details about the requirements of these fields can be found in RFC 8216bis [I-D.pantos-hls-rfc8216bis].

Attribute	Type	Required	Description
uri	string	Y	A URI that specifies how to obtain the key.
keyFormat	string	N	Specifies how the key is represented in the resource identified by the uri. If not defined, this value SHALL equal identity.
keyFormatVersions	string	N	Indicate which version(s) used in the formatting of the key this instance complies with. The value is a quoted-string containing one or more positive integers separated by the / character (for example, "1", "1/2", or "1/2/5"). If not defined, this value SHALL equal "1".
iv	string	N	A hexadecimal-sequence that specifies a 128-bit unsigned integer Initialization Vector to be used with the key.
pssh	string	N	The pssh field of a SwitchingSetProtectionSystem MUST be ignored in the case of FairPlay content protection. Additionally, a ProtectionSystemSpecificHeaderBox MUST NOT be available in the Initialization Stream of any Track belonging to this Switching Set in this case.

Table 10

3.2.10. VideoSwitchingSetType

This is the type definition of a video Switching Set.

Attribute	Type	Req?	Description
id	string	Y	The unique identifier for this Switching Set. It MUST be unique within its Presentation.
tracks	VideoTrackType[]	Y	The collection of all Tracks belonging to this Switching Set.
alignId	string	N	A unique identifier that SHOULD be set by all Switching Sets that are Aligned Switching Sets with each other.
baseUrl	string	N	The base URL of this Switching Set. It is part of the content base URLs for all Tracks belonging to this Switching Set. See Section 3.4 for more information.
codecs	string	N	The definition of the codec(s) necessary to render the content of all Tracks belonging to this Switching Set. It MUST follow the ISO File Format Name Space as defined by [RFC6381]. If this is not defined, it MUST be defined on all Tracks of this Switching Set separately.
continuationPattern	string	N	The URL pattern used to request Continuation Segments belonging to this Switching Set. The

Attribute	Type	Req?	Description
			pattern MUST include the {segmentId} string; see Section 3.4 for more information. If this is not defined, it MUST be defined on all Tracks of this Switching Set separately.
frameRate	ScaledValue	N	The frame rate of all video Tracks belonging to this Switching Set. If this is not defined, then every Track MUST set its frame rate separately.
initializationPattern	string	N	The URL pattern used to request Initialization Packets belonging to this Switching Set. The pattern MUST include the {initId} string; see Section 3.4 for more information. If this is not defined, it MUST be defined on all Tracks of this Switching Set separately.
label	string	N	A human-readable name for this Switching Set.
mediaTimeOffset	ScaledValue	N	The offset to be added to Manifest Timestamps to calculate Media Timestamps contained by Segments of this Switching Set. If neither the Track nor the Switching Set has this value set, it SHALL equal 0.
mimeType	string	N	The MIME type of all content belonging to this Switching Set. It MUST be a valid video

Attribute	Type	Req?	Description
			MIME type. If not set, the MIME type of all content of this Switching Set SHALL equal video/mp4.
protection	SwitchingSet Protection	N	The information related to content protection for all Tracks belonging to this Switching Set.

Table 11

3.2.11. MetadataSwitchingSetType

This is the type definition of a metadata Switching Set.

Attribute	Type	Req?	Description
id	string	Y	The unique identifier for this Switching Set. It MUST be unique within its Presentation.
mimeType	string	Y	The MIME type of all content belonging to this Switching Set.
tracks	MetadataTrackType[]	Y	The collection of all Tracks belonging to this Switching Set.
schemeId	string	Y	An identifier that denotes the type of metadata contained by this metadata Switching Set. Client behavior may differ depending on the vendor-specific identifier set here.
alignId	string	N	A unique identifier that SHOULD be set by all Switching Sets that are Aligned Switching Sets with each other.
baseUrl	string	N	The base URL of this Switching Set. It is part of the content base URLs for all

Attribute	Type	Req?	Description
			Tracks belonging to this Switching Set. See Section 3.4 for more information.
codecs	string	N	The definition of the codec(s) necessary to render the content of all Tracks belonging to this Switching Set. It MUST follow the ISO File Format Name Space as defined by [RFC6381]. If this Switching Set does not define it, then it must be defined on all its Tracks separately.
continuationPattern	string	N	The URL pattern used to request Continuation Segments belonging to this Switching Set. The pattern MUST include the {segmentId} string; see Section 3.4 for more information. If this is not defined, it MUST be defined on all Tracks of this Switching Set separately.
label	string	N	A human-readable name for this Switching Set.
language	string	N	The language of all Tracks of the metadata Switching Set. It MUST be specified here by its ISO 639-2 [ISO6392] code.
mediaTimeOffset	ScaledValue	N	The offset to be added to Manifest Timestamps to calculate Media Timestamps contained

Attribute	Type	Req?	Description
			by Segments of this Switching Set. If neither the Track nor the Switching Set has this value set, it SHALL equal 0.

Table 12

3.2.12. AudioTrackType

This is the type definition of an audio Track.

Attribute	Type	Req?	Description
bandwidth	uint	Y	The peak bitrate of this Track. It is denoted in bits per second. The measured bitrates of all Segments of the Track MUST NOT exceed this value. This value is used to aid in ABR decisions by the player.
id	string	Y	The unique identifier for this Track. It MUST be unique within its Switching Set.
segments	SegmentType[]	Y	The metadata of the Segments contained by this Track at the moment of Manifest creation. If a segment duration is not set, then each Segment MUST be announced by the Manifest before it is active. More information about Segment availability is given in Section 5.2.
startSegmentId	uint	N	The identifier of the first Segment in the Continuation Stream of this Track. If not defined, this value SHALL equal 0.
startSequenceNumber	uint	N	The Sequence Number of the first Initialization Packet in this Track. It

Attribute	Type	Req?	Description
			MUST denote the Sequence Number of the first Initialization Packet of this Track which was published in this Track. If not defined, this value SHALL equal 0.
averageBandwidth	uint	N	The average bitrate of this Track, denoted in bits per second. It is expected that over a duration of 10 minutes, the average bitrate of this Track SHALL be within 5% of the given value. This value is used to aid in ABR decisions by the player.
baseUrl	string	N	The base URL of this Track. It is part of the content base URLs used to request this Track's Initialization Packets and Continuation Segments. See Section 3.4 for more information.
channels	integer	N	The audio channel configuration of this audio Track, defined as the total number of audio channels in the media data. This is used for ABR selection by the player.
codecs	string	N	The definition of the codec(s) necessary to render the content of this Track. It MUST follow the ISO File Format Name Space as defined by [RFC6381]. If this is not defined here, then it MUST be defined by the Track's Switching Set.
continuationPattern	string	N	The URL pattern used to request Continuation Segments belonging to

Attribute	Type	Req?	Description
			this Track. The pattern needs to include the {segmentId} string; see Section 3.4 for more information. If this is not defined here, then it MUST be defined by the Track's Switching Set.
label	string	N	A human-readable name for this Track.
initializationPattern	string	N	The URL pattern used to request Initialization Packets belonging to this media Track. The pattern needs to include the {initId} string; see Section 3.4 for more information. If this is not defined here, then it MUST be defined by the Track's Switching Set.
mediaTimeOffset	ScaledValue	N	The offset to be added to Manifest Timestamps to calculate Media Timestamps contained by segments of this Track. If neither the Track nor the Switching Set has this value set, it SHALL equal 0.
sampleRate	uint	N	The sample rate (in Hz) of this audio Track. If this is not defined here, then it MUST be defined by the Track's Switching Set.
samplesPerFrame	uint	N	The number of audio samples in one frame of this audio Track. If neither the Track nor the Switching Set has this value set, it SHALL equal 1024.
segmentDuration	ScaledValue	N	The duration (in seconds) of each Segment contained by this Track. If this value is set, then every segmentDuration seconds,

Attribute	Type	Req?	Description
			a new Segment SHOULD be available. More information about the availability is given in Section 5.2. If not set, then each Segment MUST be individually defined by the segments field.

Table 13

For any attributes that also exist on AudioSwitchingSetType, if both AudioSwitchingSetType and AudioTrackType have a value for this attribute, then only the value set by AudioTrackType must be considered (except for the identifier and label.)

3.2.13. VideoTrackType

This is the type definition of a video Track.

Attribute	Type	Req?	Description
bandwidth	uint	Y	The peak bitrate of this Track. It is denoted in bits per second. The measured bitrates of all Segments of the Track MUST NOT exceed this value. This value is used to aid in ABR decisions by the player.
id	string	Y	The unique identifier for this Track. It MUST be unique within its Switching Set.
resolution	Resolution	Y	The resolution of this video Track.
segments	SegmentType[]	Y	The metadata of the Segments contained by this Track at the moment of Manifest creation. If a segment duration is not set, then each Segment MUST be announced by the Manifest before it is active. More information about Segment availability is given in Section 5.2.

Attribute	Type	Req?	Description
startSegmentId	uint	N	The identifier of the first Segment in the Continuation Stream of this Track. If not defined, this value SHALL equal 0.
startSequenceNumber	uint	N	The Sequence Number of the first Initialization Packet in this Track. It MUST denote the Sequence Number of the first Initialization Packet of this Track which was published in this Track. If not defined, this value SHALL equal 0.
averageBandwidth	uint	N	The average bitrate of this Track, denoted in bits per second. It is expected that over a duration of 10 minutes, the average bitrate of this Track SHALL be within 5% of the given value. This value is used to aid in ABR decisions by the player.
baseUrl	string	N	The base URL of this Track. It is part of the content base URLs used to request this Track's Initialization Packets and Continuation Segments. See Section 3.4 for more information.
codecs	string	N	The definition of the codec(s) necessary to render the content of this Track. It MUST follow the ISO File Format Name Space as defined by [RFC6381]. If this is not defined here, then it MUST be defined by the Track's Switching Set.
continuationPattern	string	N	The URL pattern used to request Continuation

Attribute	Type	Req?	Description
			Segments belonging to this Track. The pattern needs to include the {segmentId} string; see Section 3.4 for more information. If this is not defined here, then it MUST be defined by the Track's Switching Set.
frameRate	ScaledValue	N	The frame rate of this video Track. If it is not defined by the Switching Set, then it must be defined here.
label	string	N	A human-readable name for this Track.
initializationPattern	string	N	The URL pattern used to request Initialization Packets belonging to this media Track. The pattern needs to include the {initId} string; see Section 3.4 for more information. If this is not defined here, then it MUST be defined by the Track's Switching Set.
mediaTimeOffset	ScaledValue	N	The offset to be added to Manifest Timestamps to calculate Media Timestamps contained by segments of this Track. If neither the Track nor the Switching Set has this value set, it SHALL equal 0.
segmentDuration	ScaledValue	N	The duration (in seconds) of each Segment contained by this Track. If this value is set, then every segmentDuration seconds, a new Segment SHOULD be available. More information about the availability is given in Section 5.2. If not set, then each Segment MUST be

Attribute	Type	Req?	Description
			individually defined by the segments field.

Table 14

For any attributes that also exist on VideoSwitchingSetType, if both VideoSwitchingSetType and VideoTrackType have a value for this attribute, then only the value set by VideoTrackType must be considered (except for the identifier and label.)

3.2.14. Resolution

A Resolution contains the following elements:

Attribute	Type	Required	Description
width	uint	Y	The width of the picture.
height	uint	Y	The height of the picture.
sarWidth	uint	N	The width of the sample aspect ratio of the resolution. If it is not set, it SHALL equal 1.
sarHeight	uint	N	The height of the sample aspect ratio of the resolution. If it is not set, it SHALL equal 1.

Table 15

The display aspect ratio belonging to this Resolution can be calculated by using this sample aspect ratio width and height, and the picture width and height as follows:

```
dar = (darWidth, darHeight)
darWidth = sarWidth * width
darHeight = sarHeight * height
```

3.2.15. MetadataTrackType

This is the type definition of a metadata Track.

Attribute	Type	Req?	Description
id	string	Y	The unique identifier for this Track. It MUST be unique within its Switching Set.
segments	SegmentType[]	Y	The metadata of the Segments that are contained by this Track at the moment of Manifest creation. If a segment duration is not set, then each Segment

Attribute	Type	Req?	Description
			MUST be announced by the Manifest before it is active. More information about Segment availability is given in Section 5.2.
startSegmentId	uint	N	The identifier of the first Segment in the Continuation Stream of this Track. If not defined, this value SHALL equal 0.
averageBandwidth	uint	N	The average bitrate of this Track, denoted in bits per second. It is expected that over a duration of 10 minutes, the average bitrate of this Track SHALL be within 5% of the given value. This value is used to aid in ABR decisions by the player.
bandwidth	uint	N	The peak bitrate of this Track. It is denoted in bits per second. The measured bitrates of all Segments of the Track MUST NOT exceed this value. This value is used to aid in ABR decisions by the player.
baseUrl	string	N	The base URL of this Track. It is part of the content base URLs used to request this Track's Continuation Segments. See Section 3.4 for more information.
codecs	string	N	The definition of the codec(s) necessary to render the content of this Track. It MUST follow the ISO File Format Name Space as defined by [RFC6381].
continuationPattern	string	N	The URL pattern used to request Continuation Segments belonging to this Track. The pattern needs to include the {segmentId} string; see Section 3.4 for more information. If this is not defined here, then

Attribute	Type	Req?	Description
			it MUST be defined by the Track's Switching Set.
label	string	N	A human-readable name for this Track.
mediaTimeOffset	ScaledValue	N	The offset to be added to Manifest Timestamps to calculate Media Timestamps contained by segments of this Track. If neither the Track nor the Switching Set has this value set, it SHALL equal 0.
segmentDuration	ScaledValue	N	The duration (in seconds) of each Segment contained by this Track. If this value is set, then every segmentDuration seconds, a new Segment SHOULD be available. More information about the availability is given in Section 5.2. If not set, then each Segment MUST be individually defined by the segments field.

Table 16

3.2.16. SegmentType

This is the type definition of a Segment.

Attribute	Type	Required	Description
id	uint	Y	The unique identifier for this Segment within the Track. This identifier MUST be incremented by 1 for every new Segment of the same Track.
timeBounds	TimeBounds	N	The time boundaries of this Segment. If this Segment's Track does not have a constant segment duration, then this value MUST be set, and the Segment's duration MUST be available at least 2 seconds before the actual end of that Segment.

Table 17

3.2.17. PresentationEventType

Timed metadata events can be embedded in the Manifest by using Presentation Events. Each PresentationEventType contains the following elements:

Attribute	Type	Req?	Description
data	string	Y	The event payload.
id	string	Y	An unique identifier for this event. It MUST be unique within this Presentation's events.
timeBounds	PresentationEventTimeBounds	Y	The time boundaries for the event, during which the event SHALL be active.
encoding	Enum("identity", "base64", "json")	N	The content encoding of the event data. identity signifies that the encoding is plaintext. base64 signifies Base64 encoding. json signifies that the payload given by data is a valid JSON document. If this JSON document is not valid, this event MUST be ignored. If not set, the encoding SHALL default to identity.

Table 18

3.2.18. PresentationEventTimeBounds

The time boundaries of a PresentationEventType are defined as follows:

Attribute	Type	Required	Description
startTimeOffset	uint	N	The scaled start time offset of the event, defined in seconds divided by the timescale. The actual start time of the event can be calculated by dividing this value by the scale and adding the resulting value to the start time of the Presentation. It

Attribute	Type	Required	Description
			is defined as seconds divided by the timescale. If it is not set, it SHALL equal 0.
duration	uint	N	The scaled duration of the event, defined in seconds divided by the timescale. If it is not set, it SHALL equal 0.
scale	uint	N	The timescale used for these time bounds. If it is not set, it SHALL be equal to 1.

Table 19

3.3. Manifest requests

The Manifest SHOULD be available through an HTTP GET request. The URL of this Manifest can be personalized by the user.

Request path	Method	Summary
(chosen by the stream publisher)	GET	Retrieve the stream Manifest.

Table 20: Manifest Requests

3.3.1. Manifest responses

Success	
Status code	200
Content-Type	JSON (MIME type MUST equal application/vnd.theo.hesp+json)
Description	When the given Manifest exists on the server, the Manifest data is returned.

Table 21: Manifest Responses: Success

Status code	Description
404	The Manifest is not available for the given URL.

Table 22: Manifest Responses: Error

If an error occurs, the player SHOULD attempt to retry the request. In case of consecutive unsuccessful requests, the media player SHOULD assume the content is unavailable and cease playback.

3.4. Addressing of content requests

Several elements of the Manifest contain a baseUrl attribute. These attributes are used to construct the URLs of media content (and metadata.)

3.4.1. Content request URL resolution

A content request URL for a specific Track SHALL be constructed by applying relative resolution (as explained in Section 5.2 of RFC 3986 [RFC3986]) to each defined baseUrl or contentBaseUrl attribute relating to that Track, starting from the root of the Manifest. The URL of the Manifest SHALL be used as a base for the first resolution, after which the target URL of the previous resolution SHALL be used as the base URL of the next resolution.

This means that the content request URL for a Track's Initialization Stream is constructed as follows (in pseudocode):

```
T = manifestUrl
if isDefined(Manifest.contentBaseUrl):
      T = resolve(Manifest.contentBaseUrl, T)
if isDefined(Presentation.baseUrl):
      T = resolve(Presentation.baseUrl, T)
if isDefined(SwitchingSet.baseUrl):
      T = resolve(SwitchingSet.baseUrl, T)
if isDefined(Track.baseUrl):
      T = resolve(Track.baseUrl, T)
contentRequestURL = resolve(initializationPattern, T)
```

where initializationPattern is the attribute given by either SwitchingSet or Track (with Track taking precedence if given by both) and resolve(R, B) is relative resolution, where R is a URI reference and B is a base URI.

The content request URL for the Track's Continuation Stream can be constructed in the same manner if the initializationPattern is replaced with the continuationPattern. Both URLs MUST be unique for each unique Track.

An example of content addressing is given in Appendix A.1.3.

3.4.2. Requesting using an identifier and the content request URL

The content request URL MUST include the {initId} pattern for Initialization Stream URLs and the {segmentId} pattern for Continuation Stream URLs. These patterns are replaced in the actual content requests: {initId} is replaced with the requested Sequence Number of the Initialization Stream, and {segmentId} is replaced with the requested segment identifier of the Continuation Stream.

In order to add leading zeros to the identifier, the following can be added to the patterns initId and segmentId: :0(n)d where (n) is the minimal amount of characters the resulting identifier has (if the identifier is already longer, it will not be altered.)

For example, for a continuation segment with identifier 100, a following content request URL https://www.example.com/s-1/content-{segmentId:06d}.mp4 will resolve to https://www.example.com/s-1/content-000100.mp4. While the content request URL https://www.example.com/s-1/content-{segmentId:02d}.mp4 resolves to https://www.example.com/s-1/content-100.mp4

Further details about these requests are given in Section 4 and Section 5.

3.5. Manifest example

A full Manifest example, together with information on content addressing and timing information, is given by Appendix A.1.

4. Initialization Stream

The player uses the information in the HESP Manifest to fetch an Initialization Packet.

4.1. Initialization Stream purpose

The Initialization Stream is a stream containing only independent samples. This stream is not regularly used by a single client nor completely streamed. The purpose of the Initialization Stream is to make available, upon request of the client, the separate independent frames packaged in an Initialization Packet. This way, a client has more control over the specific position of the stream where it wants to initiate playback. For example, it allows for more granularity when starting playback at the live edge, seeking a specific point in time, or switching to an alternative Track.

Additionally, the Initialization Packet contains information on the position of the following frame in the Continuation Stream to achieve regular media playback.

4.2. Initialization Packet format

4.2.1. Video Initialization Packet

A video Initialization Packet MUST contain:

* A CMAF header, i.e., all information required to initialize the media decoder, stored in ISO Base Media File Format [ISOBMFF]

boxes as specified by CMAF [CMAF]. The minimal required contents are defined in Section 4.2.4

*An additional in-band metadata event containing extra HESP information, included in the aforementioned CMAF segment. The definition of such an event is denoted in Section 4.2.5.

*A CMAF segment, i.e., a group of one or more media samples, starting with at least one independent media sample, and timing information related to these samples, stored in ISOBMFF [ISOBMFF] boxes as specified by CMAF [CMAF].

4.2.2. Audio Initialization Packet

An audio Initialization Packet MUST contain:

*A CMAF header, i.e., all information required to initialize the media decoder, stored in ISO Base Media File Format [ISOBMFF] boxes as specified by CMAF [CMAF]. The minimal required contents are defined below.

*An additional in-band metadata event containing extra HESP information, included in the aforementioned CMAF segment. The definition of such an event is denoted below.

It MAY also contain an independent (audio) media sample, as is defined by the video Initialization Packet. However, it is heavily discouraged as it leads to more storage costs for the provider without any significant advantage.

4.2.3. Constraint on media information

In order to avoid any decoder anomalies, a HESP stream MUST follow the following constraint: * When the media sample(s) contained in an Initialization Packet are concatenated with the samples contained in the corresponding Continuation Segment with the referenced index, starting as of the referenced offset, the resulting bitstream should be compliant to the codec limitations.

4.2.4. CMAF header

A CMAF header is defined in ISO/IEC 23000-19:2020 [CMAF] as: "a sequence of CMAF constrained ISOBMFF boxes that do not reference any media samples (3.3.15), but are associated with a CMAF track (3.2.1) and necessary for the decoding of its CMAF fragments (3.1.1)." As such, the header MUST NOT contain any samples.

This means that for the ISO Base Media File Format, at least the following boxes MUST be given:

- An ftyp box.
- A moov box.
- One or more tracks and accompanying boxes, though stbl and other related boxes cannot give information about any actual sample entries.

4.2.5. Event message information

In order to pass information about the Continuation Stream, an in-band event in the form of an emsg box must be added to the Initialization Packet. The format of this box is defined at Section 6.2.1.1. The message_data field MUST contain the following two values (see Table 33):

- the Continuation Segment index (index), which defines the index of the Continuation Segment containing the next media sample.
- the Continuation Segment byte offset (offset), which defines the position of the next media sample in the Continuation Segment. This field is OPTIONAL if the next media sample is located at the start of the segment, in which case a byte-range request is not needed.

The message_data field of the emsg box could for example look like this:

```
{"index":1,"offset":1234}
```

4.3. Initialization Stream addressing

Section 3.4 defines how to create the correct URLs and how to request Initialization Packets. The request content URL is used together with the desired Sequence Number to retrieve an Initialization Packet.

Additionally, {initId} can also be replaced with the string now. If this URL is requested, the most recently available Initialization Packet MUST be returned.

4.3.1. Initialization Stream requests

Initialization Packets MUST be available through a basic GET request. They MUST be able to be retrieved through two methods:

Request path	Method	Summary
Initialization content request URL {initId} is replaced with the string now	GET	Retrieve the most recent Initialization Packet from the given track.
Initialization content request URL {initId} is replaced with a Sequence Number	GET	Retrieve the Initialization Packet with the given Sequence Number from the given track.

Table 23: Initialization Stream requests

4.3.2. Initialization Stream responses

Success	
Status Code	200
Content-Type	MUST match the mimeType given by the Stream's Switching Set (see Section 3 for more details)
Description	Return the requested Initialization Packet when it exists on the media server.

Table 24: Initialization Stream responses: success

Status code	Description
404	The Track does not exist on the media server, or the requested Initialization Packet does not exist on the media server.

Table 25: Initialization Stream responses: error

5. Continuation Stream

The client uses the information given by the Manifest and the Initialization Packet to request the Continuation Stream. This low latency stream is used for the bulk of media playback of HESP. The Continuation Stream is an independently playable CMAF stream.

5.1. Continuation Stream format

5.1.1. Media content

The Continuation Stream is packaged as a regular CMAF stream, albeit with some encoding constraints, depending on the chosen profile (see Appendix C.)

Maximum Gain Profile

- Samples MUST only contain references to the one sample preceding it.
- Each CMAF chunk MUST contain at most one sample (for the lowest possible latency.)
- Long CMAF Segments (values of multiple minutes are possible.)

Compatibility Profile

- Regular sized CMAF Segments (recommendation: between 1 and 30 seconds)
- Chunk sizes range between one sample and one sub-GOP
- The following GOP structure MUST be followed: I B ... B P B ... B P B ... B P
- The number of B frames MAY vary from 0 (same behavior as Maximum Gain Profile) to 4 (recommended maximum)
- Each sub-GOP (B ... B P) MUST only reference one previous frame (allowing injection of keyframes)

5.2. Continuation Segment availability

Continuation Segments must together create a continuous stream of media data. There MUST NOT exist gaps in the time boundaries of subsequent Continuation Segments.

There exist two ways to signal Segment availability. Either the Track has a segmentDuration set in the Manifest, which represents the constant duration of each Segment. In this case, a client can derive on its own when to start requesting a new Segment. A new Segment SHOULD be published every n seconds, where n equals segmentDuration. This new Segment MUST become available within 100 milliseconds from that point. It is not allowed to drift, i.e., if a Segment is only published at n+100ms, then the next Segment MUST be available at 2n+100ms at the latest.

The duration of the last Segment of a Track MAY be shorter than this constant segmentDuration. As a new Manifest must be retrieved near the end of a Presentation, a client should be able to start requesting content of the new Presentation in a timely manner, regardless of the length of this last Segment.

The other option is for the Track to specify all its Segments outright in the segments field of the Manifest. As the Manifest does not get regularly retrieved, it is REQUIRED to signal a Manifest update (see Section 6.2.1.2) at the end of each Segment. It is recommended only to use this option in the "Maximum Gain Profile", as it can otherwise significantly increase the number of Manifest requests necessary.

5.3. Continuation Stream addressing

To start playback of a Track, a client MUST request the Continuation Segments of the chosen Track(s) using the information given by (already obtained) the Manifest and Initialization Packet.

Requests for Content Segments should be made using HTTP GET requests. The first request, following an Initialization Packet, fetches the Segment indicated by the Initialization Packet. A byte-range header should be used to specify the range of data that needs to be requested. This information is also given by the Initialization Packet.

A client then automatically calculates the URL of the next Segment as it is indicated in the Manifest and requests the next Segment using a regular HTTP GET request.

5.3.1. Continuation Stream URLs

Section 3.4 defines how to create the correct URLs and how to request a continuation segment. The request content URL is used together with the desired segment identifier to retrieve the Segment.

5.3.2. Continuation Stream requests

Suppose a client needs to request a partial Continuation Segment, for example, starting at the byte offset given by metadata of an Initialization Packet. In that case, it MUST use one or more HTTP Range [RFC7233] Requests. The start and end of the range of each request MUST be defined.

Often, the total length of the Continuation Segment will not yet be known at the time of the request. If a client cannot define an accurate end value for the HTTP Range of a request, then 2^53 - 1 (9007199254740991) SHOULD be used.

Request path	Method	Summary
Continuation content request URL {segmentId} is replaced with a segment identifier	GET	Retrieve the Continuation Segment with the requested identifier from the given track.

Table 26: Continuation Stream requests

5.3.3. Continuation Stream responses

A distinction is made between how responses should be returned, depending on the version of the HTTP protocol used.

5.3.3.1. HTTP/1.1 successful response

If an HTTP Range request is sent, then a 206 Partial Content response MUST be returned upon a successful response. The response MUST use Chunked Transfer Coding [RFC7230] to ensure timely delivery of media data.

Success	
Status Code	206
Content-Type	MUST match the mimeType given by the Stream's Switching Set (see Section 3 for more details)
Transfer-Encoding	chunked
Description	The requested range of Segment data is returned from the server. Depending on the byte-range requested, the connection is kept open to retrieve live data.

Table 27: Continuation Stream responses: success (Range header is given)

For requests without a Range header, a 200 OK response MUST be returned upon success.

Success	
Status Code	200
Content-Type	MUST match the mimeType given by the Stream's Switching Set (see Section 3 for more details)
Transfer-Encoding	chunked
Description	The Segment data is returned from the server. Depending on the availability of the Segment, the connection is kept open to retrieve live data.

Table 28: Continuation Stream responses: success (Range header is not given)

5.3.3.2. HTTP/2 successful response

HTTP/2 uses frame-based transmission and cannot use Chunked Transfer Coding. As such, this header is not given here.

Success	
Status code	206
Content-Type	MUST match the mimeType given by the Stream's Switching Set (see Section 3 for more details)
Description	The requested range of Segment data is returned from the server. Depending on the byte-range requested, the connection is kept open to retrieve live data.

Table 29: Continuation Stream responses: success (Range header is given)

For requests without a Range header, a 200 OK response MUST be returned upon success.

Success	
Status code	200
Content-Type	MUST match the mimeType given by the Stream's Switching Set (see Section 3 for more details)
Description	The Segment data is returned from the server. Depending on the availability of the Segment, the connection is kept open to retrieve live data.

Table 30: Continuation Stream responses: success (Range header is not given)

5.3.3.3. Response errors

Status code	**Description**
404	The Track does not exist on the media server, or the requested Segment does not exist on the media server.
416	The requested byte-range could not be fulfilled.

Table 31: Continuation Stream responses: error

6. Timed metadata

HESP supports timed metadata through two methods: metadata Tracks for continuous, often segmented metadata and metadata events for sporadic metadata updates.

6.1. Metadata Tracks

Metadata Tracks function similarly to media Tracks, however, without the need for an Initialization Stream. The attributes for metadata Tracks in the Manifest can be found at Section 3.2.15. The addressing happens similarly to the addressing of Continuation segments: a continuationPattern is given for each metadata Track,

either directly set on the Track or on the Switching Set of the Track. Each metadata Segment contains a numerical identifier that increments by one for each new Segment of the Track. This identifier, together with the addressing pattern, creates the URL used to retrieve the contents of the Segment.

The Manifest contains the duration of each metadata Segment, either stated individually per Segment or through the segmentDuration attribute set on the metadata Track. It is possible that chunked encoding is used here to ensure that the contents are delivered as soon as possible, but this is not a requirement. This can be useful for subtitles alongside live content, for example.

6.2. Metadata events

Metadata events can be used to signal information that does not become available in regular intervals. These events can either be transmitted in-band, where it MUST be added to the video or audio Continuation Streams, or out-of-band, in which case details about the metadata event MUST be available in the Manifest.

6.2.1. In-band events

In order to deliver events in-band, root-level Event Message (emsg) boxes MUST be added to ongoing media Continuation Streams. The definition of an emsg box can be found in the CMAF [CMAF] specification.

These boxes should be appended to each Track of a Presentation to ensure all viewers can receive such an event. It is recommended to use out-of-band events if the included data is significantly large.

The HESP specification defines a few in-band events that it leverages for client initialization and Manifest updates. The structure of these events is given below.

6.2.1.1. Initialization data

This event is appended to each Initialization Packet for the client to request the correct Continuation Segments. It is an ´(U+00B4)emsg´ (U+00B4) box with the following REQUIRED values:

Attribute	Value
version	0
scheme_id_uri	"urn:theo:hesp:2020"
value	"initdata"
timescale	MUST match the timescale of the Initialization Packet in case of video or MUST equal 1 otherwise.

Attribute	Value
presentation_time_delta	0
event_duration	MUST match the duration of the Initialization Packet in case of video or MUST equal 0 otherwise.
id	can be freely set (and MUST be ignored by the player)
message_data	MUST contain the data defined by Table 33, formatted as JSON

Table 32: emsg box containing an initialization data event

Attribute	Type	Required	Description
index	integer	Y	The Continuation Segment index (see Section 4.2.5.)
offset	integer	N	The Continuation Segment byte offset (see Section 4.2.5), it SHALL be 0 if not given here.

Table 33: message_data contents of an initialization data event

6.2.1.2. Manifest update

This event is used to signal the client that a new Manifest must be retrieved. This can occur for many reasons, such as before a Presentation change, availability of new out-of-band metadata, etc. It is an emsg box with the following REQUIRED values:

Attribute	Value
version	0
scheme_id_uri	"urn:theo:hesp:2020"
value	"manifestupdate"
timescale	1
presentation_time_delta	0
event_duration	0
id	MAY be freely set (and MUST be ignored by the player)
message_data	MUST contain the data defined by Table 35, formatted as JSON

Table 34: emsg box containing a Manifest update event

Attribute	Type	Required	Description
url	string	N	the URL of an alternative location of the Manifest

Table 35: message_data contents of a Manifest data event

If the event contains a URL, then this Manifest request MUST be made to this address. Further Manifest requests do not have this requirement.

6.2.2. Out-of-band events

Out-of-band events can be added to the Manifest through Presentation Events. The definition of a Presentation Event is given in Section 3.2.17. The contents of the event data are not predefined. It is possible to include arbitrary data as Base64 encoded data, and plaintext data can be included as-is. If needed, the data can be an URL that needs to be resolved separately, but this is up to the publisher of this data.

7. Content protection

HESP has support for content protection. It allows DRM systems to be implemented through the common encryption standard.

7.1. Common encryption support

As HESP has the requirement for media to be structured as ISOBMFF [ISOBMFF], common encryption [CENC] is to be used to encrypt that media content.

Common encryption specifies 4 protection schemes that may be used: cenc, cbc1, cens and cbcs. For HESP, either the AES-CBC subsample pattern encryption scheme (cbcs) or the AES-CTR full sample pattern encryption scheme (cenc) MUST be used.

7.2. HESP Manifest

The HESP Manifest has a SwitchingSetProtection structure to set up common encryption for audio and video Switching Sets. The definition of this structure can be found in Chapter 3. As a result of this being set on a Switching Set level:

* All Tracks belonging to such a Switching Set MUST be encrypted with the same content key. Aligned Switching Sets can be used to ensure that a client can still switch through Tracks of different Switching Sets.
* Audio and video data SHOULD be encrypted with different content keys. This is a recommendation as both often have different encryption strength requirements.

The SwitchingSetProtection structure MAY contain a ProtectionSystemSpecificHeaderBox (pssh), which can also be contained by the Initialization Stream. Note that this box MUST be given by at least one of both options in order for a license request to be made. If both exist for a Switching Set, the pssh box from the

Initialization Stream MUST be disregarded. This allows for more straightforward alterations of license information after a stream has been created or published.

7.3. CMAF box structure

The media of a protected stream needs to contain certain ISOBMFF boxes to be compliant with CENC (and CMAF). HESP requires the same boxes to be present in media streams. A brief overview of these boxes is given below. More information on this requirement can be found in the CENC [CENC] and CMAF [CMAF] specifications.

7.3.1. Initialization Stream

Initialization Packets MUST contain the following boxes:

* TrackEncryptionBox (tenc): This box contains default parameters regarding sample encryption.
* SchemeTypeBox (schm): This box identifies the protection scheme. scheme_type MUST be set to cbcs or cenc.
* ProtectionSystemSpecificHeaderBox (pssh): If the Manifest does not contain a pssh box that applies to this Track, then this box MUST be included in each Initialization Packet of each Track of the protected Switching Set. If the Manifest does contain a pssh box that applies to this Track, then this box MUST be disregarded.

In order to signal that the Track is encrypted, the stream type MUST equal encv for video Tracks and enca for audio Tracks.

Additionally, video Initialization Packets contain a sample. If that sample is encrypted, then the same requirements for CMAF fragments of a Continuation Segment also apply here: a senc box MUST be included, pssh, saiz and saio boxes MAY be included. The following section contains more information about these requirements.

7.3.2. Continuation Stream

CMAF fragments of a Continuation Segment MUST contain the following box:

* SampleEncryptionBox (senc): This box is used to store initialization vector data and information on subsample encryption. As each chunk in HESP must currently contain at most one sample, sample_count SHALL always 1.

The following boxes MAY be included:

- ProtectionSystemSpecificHeaderBox (pssh): If any updates need to be made to the underlying licensing system, a pssh box MAY be included.
- SampleAuxiliaryInformationSizesBox (saiz): This box is used to store the size of per-sample auxiliary information. It is only REQUIRED if such per-sample information exists.
- SampleAuxiliaryInformationOffsetsBox (saio): This box is used to store the offsets of per-sample auxiliary information. It is only REQUIRED if such per-sample information exists.

8. Contributors

Significant contributions to the design of this protocol were made by Egon Okerman, Samie Beheydt, and Johan Vounckx.

9. IANA Considerations

This memo requests that the following MIME type [RFC2046] be registered with the IANA:

Type name: application

Subtype name: vnd.theo.hesp+json

Required parameters: (none)

Optional parameters: (none)

Encoding considerations: encoded as text.

Security considerations: See Section 10.

Compression: this media type does not employ compression.

Interoperability considerations: There are no byte-ordering issues, since files are 7- or 8-bit text. Applications could encounter unrecognized tags, which SHOULD be ignored.

10. Security Considerations

Since the protocol uses HTTP to transmit data, the regular HTTP security considerations apply. See section 15 of RFC 7230 [RFC7230].

Clients SHOULD take care when parsing files received from a server so that non-compliant files are rejected. Clients SHOULD range-check responses to prevent buffer overflows. See also the Security Considerations section of RFC 3986 [RFC3986]. Clients SHOULD load

resources identified by URI lazily to avoid contributing to denial-of-service attacks.

HTTP requests often include session state ("cookies"), which may contain private user data. Implementations MUST follow cookie restriction and expiry rules specified by RFC 6265 [RFC6265]. See also the Security Considerations section of RFC 6265, and RFC 2964 [RFC2964].

Encryption keys are specified by URI. The delivery of these keys SHOULD be secured by a mechanism such as HTTP over TLS [RFC8446] (formerly SSL) in conjunction with a secure realm or a session cookie.

11. Normative References

[CENC] International Organization for Standardization, "Information technology - MPEG systems technologies - Part 7: Common encryption in ISO base media file format files", December 2020, <https://www.iso.org/standard/68042.html>.

[CMAF] International Organization for Standardization, "Information technology - Multimedia application format (MPEG-A) - Part 19: Common media application format (CMAF) for segmented media", March 2020, <https://www.iso.org/standard/79106.html>.

[DASH] International Organization for Standardization, "Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats", December 2019, <https://www.iso.org/standard/79329.html>.

[I-D.pantos-hls-rfc8216bis] Pantos, R., "HTTP Live Streaming 2nd Edition", Work in Progress, Internet-Draft, draft-pantos-hls-rfc8216bis-12, 11 November 2022, <https://datatracker.ietf.org/doc/html/draft-pantos-hls-rfc8216bis-12>.

[ISO6392] International Organization for Standardization, "Codes for the representation of names of languages - Part 2:

Alpha-3 code", November 1998, <https://www.iso.org/standard/4767.html>.

[ISO8601] International Organization for Standardization, "Date and time - Representations for information interchange", February 2019, <https://www.iso.org/standard/70907.html>.

[ISOBMFF] International Organization for Standardization, "Information technology - Coding of audio-visual objects - Part 12: ISO base media file format", December 2020, <https://www.iso.org/standard/74428.html>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2964] Moore, K. and N. Freed, "Use of HTTP State Management", BCP 44, RFC 2964, DOI 10.17487/RFC2964, October 2000, <https://www.rfc-editor.org/info/rfc2964>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <https://www.rfc-editor.org/info/rfc6265>.

[RFC6381] Gellens, R., Singer, D., and P. Frojdh, "The 'Codecs' and 'Profiles' Parameters for "Bucket" Media Types", RFC 6381, DOI 10.17487/RFC6381, August 2011, <https://www.rfc-editor.org/info/rfc6381>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7233] Fielding, R., Ed., Lafon, Y., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Range Requests", RFC 7233, DOI 10.17487/RFC7233, June 2014, <https://www.rfc-editor.org/info/rfc7233>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[UUID] International Organization for Standardization, "Information technology - Procedures for the operation of object identifier registration authorities - Part 8: Generation of universally unique identifiers (UUIDs) and their use in object identifiers", August 2014, <https://www.iso.org/standard/62795.html>.

12. Informative References

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

Appendix A. Example usage

A.1. Manifest

For the initial step, the client retrieves a Manifest.

A.1.1. Retrieving the Manifest

The URL of the Manifest is given out of band to the client. The client sends out a GET request. In this case, let's suppose the Manifest is available at https://example.com/stream1/manifest.json. The client then makes such a request:

```
GET /stream1/manifest.json HTTP/1.1
Host: example.com
Accept: application/vnd.theo.hesp+json
```

The server responds with the following headers:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.theo.hesp+json; charset=utf-8
Content-Length: 6867
Date: Wed, 31 Mar 2021 08:00:00 GMT
```

and the following body:

```
{
   "activePresentation":"1",
   "availabilityDuration":{
      "value":2400
   },
   "creationDate":"2021-03-31T08:00:00.000Z",
   "fallbackPollRate":300,
   "manifestVersion":"2.0.0",
   "streamType":"live",
   "currentTime":{
            "value":1134000000,
            "scale":90000
   },
   "presentations":[
      {
         "id":"0",
         "timeBounds":{
            "startTime":0,
            "endTime":972000000,
            "scale":90000
         },
         "audio":[
            {
               "id":"main-audio",
               "language":"eng",
               "baseUrl":"audio/",
               "channels":2,
               "codecs":"mp4a.40.2",
               "continuationPattern":"content-{segmentId}.mp4",
               "initializationPattern":"init-{initId}.mp4",
               "sampleRate":48000,
               "tracks":[
                  {
                     "id":"96kbps",
                     "averageBandwidth":96000,
                     "bandwidth":96000,
                     "baseUrl":"96k/",
                     "segmentDuration":{
                        "value":540000,
                        "scale":90000
                     },
                     "segments":[
                        {
                           "id":1799,
                           "timeBounds":{
                              "startTime":971460000,
                              "scale":90000
                           }
                        }
```

```
                ]
             }
          ]
       }
    ],
    "video":[
       {
          "id":"main-video",
          "baseUrl":"video/",
          "frameRate":{
             "value":25
          },
          "continuationPattern":"content-{segmentId}.mp4",
          "initializationPattern":"init-{initId}.mp4",
          "tracks":[
             {
                "id":"720p",
                "bandwidth":3000000,
                "baseUrl":"720p/",
                "codecs":"avc1.4d001f",
                "resolution":{
                   "width":1280,
                   "height":720
                },
                "segmentDuration":{
                   "value":540000,
                   "scale":90000
                },
                "segments":[
                   {
                      "id":1799,
                      "timeBounds":{
                         "startTime":971460000,
                         "scale":90000
                      }
                   }
                ]
             }
          ]
       }
    ]
 },
 {
    "id":"1",
    "timeBounds":{
       "startTime":972000000,
       "scale":90000
    },
    "baseUrl":"https://otherexample.com/s2/",
```

```
"audio":[
   {
      "id":"main-audio",
      "language":"eng",
      "baseUrl":"audio/",
      "channels":2,
      "codecs":"mp4a.40.2",
      "sampleRate":48000,
      "mediaTimeOffset":{
         "value":-972000000,
         "scale":90000
      },
      "tracks":[
         {
            "id":"128kbps",
            "averageBandwidth":128000,
            "bandwidth":128000,
            "continuationPattern":
               "128k-content-{segmentId}.mp4",
            "initializationPattern":
               "128k-init-{initId}.mp4",
            "segmentDuration":{
               "value":540000,
               "scale":90000
            },
            "segments":[
               {
                  "id":300,
                  "timeBounds":{
                     "startTime":1134000000,
                     "scale":90000
                  }
               }
            ]
         }
      ]
   }
],
"video":[
   {
      "id":"main-video",
      "baseUrl":"video/",
      "frameRate":{
         "value":25
      },
      "mediaTimeOffset":{
         "value":-972000000,
         "scale":90000
      },
```

```
"tracks":[
   {
      "id":"720p",
      "bandwidth":3000000,
      "codecs":"avc1.4d001f",
      "continuationPattern":
         "720p-content-{segmentId}.mp4",
      "initializationPattern":
         "720p-init-{initId}.mp4",
      "resolution":{
         "width":1280,
         "height":720
      },
      "segmentDuration":{
         "value":540000,
         "scale":90000
      },
      "segments":[
         {
            "id":300,
            "timeBounds":{
               "startTime":1134000000,
               "scale":90000
            }
         }
      ]
   },
   {
      "id":"1080p",
      "bandwidth":5000000,
      "codecs":"avc1.4d001f",
      "continuationPattern":
         "1080p-content-{segmentId}.mp4",
      "initializationPattern":
         "1080p-init-{initId}.mp4",
      "resolution":{
         "width":1920,
         "height":1080
      },
      "segmentDuration":{
         "value":540000,
         "scale":90000
      },
      "segments":[
         {
            "id":300,
            "timeBounds":{
               "startTime":1134000000,
               "scale":90000
```

```
                        }
                    }
                ]
            }
        ]
    }
  ]
}
```

A.1.2. Timing information

In the above Manifest, we have two Presentations:

* The first Presentation with ID "0" is not active at the current time, though it was active previously. It contains one audio Track and one video Track. It is 3 hours long in total and starts at 00:00:00.000 (Manifest Timestamp.) No Track has a mediaTimeOffset defined, so the Manifest Timestamps match the Media Timestamps.
* The second Presentation with ID "1" is active at the current time. It contains one audio Track and two video Tracks. Its total duration is yet to be defined. Its start time is 03:00:00.000 in Manifest Time and its current time (at the moment the Manifest was retrieved) is 03:30:00.000. All Tracks have the same mediaTimeOffset defined. The Media Timestamps for this Presentation start at 00:00:00.000. That also means that for all the currently active Segments, the media data will contain a starting timestamp of 00:30:00.000.

The availabilityDuration of the Manifest is 40 minutes. As only 30 minutes of the second Presentation have elapsed, some Segments of the first Presentation are still available. Therefore, the first Presentation must still be included in the Manifest. Once 10 minutes have passed, the first Presentation can be left out of the Manifest.

A.1.3. Content addressing

The client uses the Manifest to derive the content request URLs of each Track.

For the audio Track of the first Presentation, the following parts are found:

```
AudioSwitchingSet.baseUrl = "audio/"
AudioTrack.baseUrl = "96k/"
AudioSwitchingSet.initializationPattern = "init-{initId}.mp4"
AudioSwitchingSet.continuationPattern = "content-{segmentId}.mp4"
```

The Manifest URL (https://example.com/stream1/manifest.json) is used as the base for resolution, and then relative resolution is applied to each of the parts above.

```
T = resolve(AudioSwitchingSet.baseUrl, manifestUrl)
  = "https://example.com/stream1/audio/"
T = resolve(AudioTrack.baseUrl, T)
  = "https://example.com/stream1/audio/96k/"
initBaseUrl = resolve(AudioSwitchingSet.initializationPattern, T)
  = "https://example.com/stream1/audio/96k/init-{initId}.mp4"
contBaseUrl = resolve(AudioSwitchingSet.continuationPattern, T)
  = "https://example.com/stream1/audio/96k/content-{segmentId}.mp4"
```

The final content request URLs are https://example.com/stream1/audio/96k/init-{initId}.mp4 and https://example.com/stream1/audio/96k/content-{segmentId}.mp4 for the Initialization Stream and Continuation Stream respectively.

For the audio Track of the second Presentation, the following parts are found:

```
Presentation.baseUrl = "https://otherexample.com/s2/"
AudioSwitchingSet.baseUrl = "audio/"
AudioTrack.initializationPattern = "128k-init-{initId}.mp4"
AudioTrack.continuationPattern = "128k-content-{segmentId}.mp4"
```

The Manifest URL (https://example.com/stream1/manifest.json) is used as the base for resolution, and then relative resolution is applied to each of the parts above.

```
T = resolve(Presentation.baseUrl, manifestUrl)
  = "https://otherexample.com/s2/"
T = resolve(AudioSwitchingSet.baseUrl, T)
  = "https://otherexample.com/s2/audio/"
initBaseUrl = resolve(AudioTrack.initializationPattern, T)
  = "https://otherexample.com/s2/audio/128k-init-{initId}.mp4"
contBaseUrl = resolve(AudioTrack.continuationPattern, T)
  = "https://otherexample.com/s2/audio/128k-content-{segmentId}.mp4"
```

The final content request URLs are https://otherexample.com/s2/audio/128k-init-{initId}.mp4 and https://otherexample.com/s2/audio/128k-content-{segmentId}.mp4 for the Initialization Stream and Continuation Stream respectively.

A.2. Initialization Stream

In the second step, the client uses the Manifest to retrieve and then parse an Initialization Packet.

A.2.1. Retrieving Initialization Packets

The client decides to retrieve the only audio Track of the active Presentation. The URL pattern for the Initialization Stream of this Track is https://otherexample.com/s2/audio/128k-init-{initId}.mp4.

The client opts to retrieve the most recent Initialization Packet and sends out a request:

```
GET /s2/128k-init-now.mp4 HTTP/1.1
Host: otherexample.com
Accept: */*
```

The server responds with the following headers

```
HTTP/1.1 200 OK
Content-Type: audio/mp4
Content-Length: 742
Date: Wed, 31 Mar 2021 08:00:02 GMT
```

and a binary body containing the ISOBMFF media data of the audio Initialization Packet.

The client repeats this to retrieve one of the video Tracks.

A.2.2. Parsing offset information

The client parses the ISOBMFF boxes in the audio Initialization Packet. Once the emsg box with scheme ID urn:theo:hesp:2020 and value initdata is found, its message_data field is parsed.

In this case, message_data equals {"index":200,"offset":63275}. The following media data for the chosen audio Track is available in the Continuation Segment with ID 200 and at a byte offset of 63275.

A.3. Continuation Stream

In the final step, the client uses the retrieved information to retrieve Continuation Segments and reaches playback.

A.3.1. Retrieving Continuation Segments

The client retrieves the Continuation Segment of the chosen audio Track. The URL pattern for the Continuation Stream of this Track is https://otherexample.com/s2/audio/128k-content-{segmentId}.mp4.

The client sends out a request for the segment with ID 200, starting at offset 63275:

```
GET /s2/128k-content-200.mp4 HTTP/1.1
Host: otherexample.com
Accept: */*
Range: bytes=63275-9007199254740991
```

As discussed in Section 5.3.2, a sufficiently large end byte value is chosen to ensure the entire range is retrieved.

The server responds with the following headers

```
HTTP/1.1 200 OK
Content-Type: audio/mp4
Content-Range: bytes 63275-9007199254740991/9007199254677716
Transfer-Encoding: chunked
Date: Wed, 31 Mar 2021 08:00:03 GMT
```

and a (chunked) binary body containing the ISOBMFF media data of the Continuation Segment.

The client repeats this for the chosen video Track and uses the retrieved media data to reach playback. The Manifest that was retrieved previously contains sufficient information to retrieve new Continuation Segments from this point on.

Appendix B. CDNs

A Content Delivery Network (CDN) MAY be employed to increase the scalability and cacheability for delivering HESP.

While the HESP protocol uses HTTP/1.1 and delivery should be possible over most HTTP CDNs, care must be taken to ensure that the CDN has all the required features.

To correctly handle HESP, the CDN MUST either support HTTP/1.1 with chunked transfer encoding or support HTTP/2. It MUST also support Range Requests as well as the caching of the partial object responses. This ensures that HESP requests and responses pass correctly through the CDN and that responses can be cached for future use.

The CDN SHOULD support collapsing multiple HTTP/1.1 Range Requests with overlapping byte-ranges into a single request. This ensures that two requests with byte-ranges that are partially overlapping require only a single request to the media server for the overlapping part. This reduces the number of concurrent requests arriving on the media server.

Appendix C. HESP Profiles (using H.264 as video codec)

In this annex, we describe two possible profiles for video Tracks of HESP streams. Both profiles place certain requirements on the underlying H.264 encoding of the HESP stream.

C.1. Maximal Gain Profile

The goal of this profile is to allow the stream to reach the lowest latency and zapping times possible using the HESP protocol. In this profile, it should be ensured that an Initialization Packet exists

for each frame of the Continuation Stream. As a result, a client can start playback, seek and switch between Tracks at any time position of the stream.

The Initialization Stream for Tracks of the Maximal Gain Profile must satisfy the following requirements:

- The frame rate of the Initialization Stream MUST match the Continuation Stream.
- Each media sample of the Initialization Stream MUST be independent (i.e., an I frame in H.264) and individually addressable (the latter is currently always true for HESP.)

The Continuation Stream for Tracks of the Maximal Gain Profile must satisfy the following requirements:

- Each media sample of the Continuation Stream MUST be either independent (i.e., an I frame in H.264) or dependent on the media sample directly preceding it in decode order (i.e., a P frame referencing only the previous frame.)
- Each CMAF Fragment of the Continuation Segment MUST only contain one media sample.
- Each Continuation Segment SHOULD be significantly long (values of multiple minutes are possible and encouraged.)

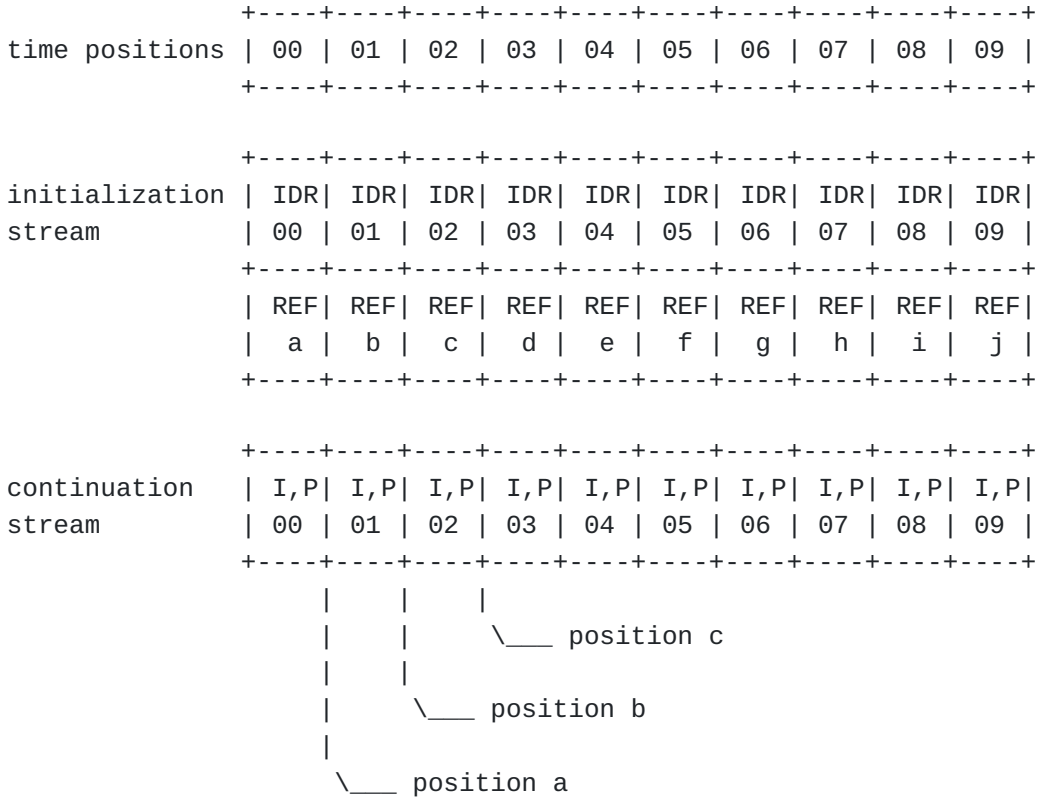


Figure 4: Maximum Gain Profile

C.2. Compatibility Profile

The goal of this profile is to allow media data from other HTTP-based adaptive bitrate protocols to be reused. This comes at the cost of some optimizations made by the previous profile. In this profile, it is not possible to start playback at any time position of the stream, as it is not possible for Initialization Packets to refer to every sample of a Continuation Segment.

The Initialization Stream for Tracks of the Compatibility Profile must satisfy the following requirements:

* Each media sample of the Initialization Stream MUST be independent (i.e., an I frame in H.264) and individually addressable (the latter is currently always true for HESP.)
* Initialization Packets MUST contain a reference to the subsequent sample in the Continuation Segment, where this subsequent sample MUST be either independent (i.e., an I frame in H.264) or dependent on the media sample directly preceding it in decode order (i.e., a P frame referencing only the previous frame.) If the subsequent sample does not meet this constraint, then this Initialization Packet MUST NOT be published. Instead, the last valid Initialization Packet MUST be returned if this Sequence Number is queried.

The Continuation Stream for Tracks of the Compatibility Profile must satisfy the following requirements:

* Continuation Segments SHOULD NOT exceed a duration of 30 seconds.
* Each CMAF Fragment of the Continuation Segment MUST contain at most the amount of media samples of a sub-GOP (defined below.)
* The following GOP structure must be followed in the underlying H.264 stream: I, B (repeated n times), P, B (repeated n times), P, where n lies between 0 and 4.
* Each sub-GOP (B ... B P) MUST depend on at most one previous frame (allowing for keyframe insertion.)

C.2.1. Example

Figure 5 depicts a part of H.264 output of 7 frames, sorted in decode order. The dependencies of each frame are shown with arrows.

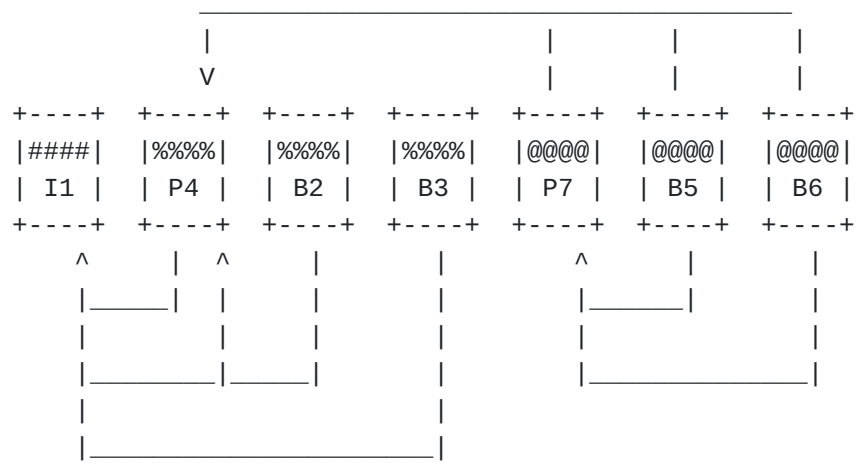


Figure 5: Continuation stream with sub-GOPs

C.2.1.1. Sub GOPs

A "sub-GOP" defines a set of B and P frames that only depend on one previous frame.

In Figure 5, there are 3 sub-GOPs:

- sub-GOP 1 (####) contains only a single I frame.
- sub-GOP 2 (%%%%) contains a P frame (P4) that only depends on I1; all B frames depend on I1 and P4.
- sub-GOP 3 (@@@@) contains a P frame (P7) that only depends on P4; all B frames depend on P7 and P4.

C.2.1.2. Initialization Packets

An Initialization Packet can be published if the subsequent media sample of the Continuation Stream depends on at most one previous frame.

For Figure 5, this is the case at the following positions:

- position 1: an Initialization Packet can be published. It will contain IDR1 (a keyframe matching the timestamp of I1) and will reference P4, the subsequent media sample of the Continuation Stream that only depends on I1. On the client-side, the media data will be decoded with IDR1 inserted at the location of I1.
- position 4: an Initialization Packet can be published. It will contain IDR4 (a keyframe matching the timestamp of P4) and will reference P7, the subsequent media sample of the Continuation Stream that only depends on P4. On the client-side, the media data will be decoded with IDR4 inserted at the location of P4.

A client requesting an Initialization Packet at other time positions must receive the most recent valid Initialization Packet. For

example, that means that a request for an Initialization Packet at position 2 in Figure 5 must return the Initialization Packet at position 1.

Author's Address

Pieter-Jan Speelmans (editor)
THEO Technologies
Leuven
Belgium

Email: pieter-jan.speelmans@theoplayer.com