Consumer Technology Association™

CTA Specification

Device Playback Capabilities Specification

CTA-5003-A

September 2023

NOTICE

(Formulated under the cognizance of the CTA **WAVE Project**; for information please see cta.tech/WAVE.)

Published by
CONSUMER TECHNOLOGY ASSOCIATION
Technology & Standards Department
www.cta.tech

WAVE

WEB APPLICATION VIDEO ECOSYSTEM

Device Playback Capabilities Specification

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

FOREWORD

This document was developed by the Web Application Video Ecosystem (WAVE) Project of the Consumer Technology Association. The WAVE Project is a broad industry initiative of content, technology, infrastructure and device companies, all working together towards commercial Internet video interoperability based on industry standards.

WAVE Device Playback Capabilities Specification

1 SCOPE

The scope of this document is to define normative requirements around playback of CTA WAVE content, i.e., primarily segmented media content. These requirements will be applicable to HTML-5 based playback of type 1 and type 3 as well as non-HTML5 devices. Playback of content includes detecting the ability to playback WAVE content and programs (where WAVE programs are sequences of CMAF presentations), playing back the content in different scenarios (regular, random access, chunked mode, etc.).

2 REFERENCES

2.1 Normative References

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

These normative references are intended to include corrigenda and amendments available at the time of use.

[CMAF]	ISO/IEC 23000-19, *Information technology — Coding of audio-visual objects — Part 19: Common media application format (CMAF) for segmented media,* https://www.iso.org/standard/85623.html.
[DASH]	ISO/IEC 23009-1, *Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats,* https://www.iso.org/standard/83314.html.
[ETSI AC-3]	ETSI TS 102 366, *Digital Audio Compression (AC-3, Enhanced AC-3) Standard,* https://www.etsi.org/standards.
[ETSI AC-4]	ETSI TS 103 190-2, *Digital Audio Compression (AC-4) Standard, Part 2: Immersive and personalized audio,* https://www.etsi.org/standards.
[ISO/IEC CICP]	ISO/IEC 23091-3:2018: *Information technology — Coding-independent code points — Part 3: Audio*, https://www.iso.org/standard/73413.html.
[ITU SYNC]	Rec. ITU-R BT.1359-1, *Relative Timing for Sound and Vision for Broadcasting*, https://www.itu.int/pub/R-REC.
[W3C MCAP]	W3C Media Capabilities, Editor's Draft 17 November 2022, https://w3c.github.io/media-capabilities/.

[W3C MSE] W3C Recommendation: “Media Source Extensions”, https://www.w3.org/TR/media-source/.

[W3C EME] Encrypted Media Extensions, W3C Recommendation 21 September 2022, https://www.w3.org/TR/encrypted-media/.

[WAVE-CON] CTA-5001-E, *Web Application Video Ecosystem (WAVE) Content Specification*, 2022 Edition, https://www.cta.tech/Resources/Standards.

[WAVE-WMA] Web Media API Snapshot 2022, December 2022, https://w3c.github.io/webmediaapi/.

2.2 Informative References

The following documents contain provisions that, through reference in this text, constitute informative provisions of this document. At the time of publication, the editions indicated were valid. All documents are subject to revision, and parties to agreements based on this document are encouraged to investigate the possibility of applying the most recent editions of the documents listed here.

[ECMASCRIPT-5.1] Ecma-262 Edition 5.1, *ECMAScript Language Specification*, Ecma International. June 2011. https://www.ecma-international.org/publications-and-standards/standards/.

[MEDIA-SOURCE] Media Source Extensions, W3C Recommendation 17 November 2016, http://www.w3.org/TR/media-source/

[MSE-FORMAT-ISOBMFF] ISO BMFF Byte Stream Format, W3C Working Group Note 04 October 2016, http://www.w3.org/TR/mse-byte-stream-format-isobmff/

[HTML51] W3C HTML 5.1 2nd Edition. Steve Faulkner; Arron Eicholz; Travis Leithead; Alex Danilo. W3C Recommendation 3 October 2017. https://www.w3.org/TR/html51/

[WEBAUDIO] Web Audio API. Paul Adenot; Chris Wilson; Chris Rogers. W3C. 8 December 2015. W3C Working Draft. https://www.w3.org/TR/webaudio/

3 DOCUMENT NOTATION AND CONVENTIONS

The following terms are used to specify conformance elements of this specification. These are adopted from the ISO/IEC Directives, Part 2, Annex H [ISO-P2H ISO-P2H]. For more information, please refer to those directives.

- MUST and MUST NOT indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.
- SHOULD and SHOULD NOT indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required, or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.
- MAY and NEED NOT indicate a course of action permissible within the limits of the document.

Terms defined to have a specific meaning within this specification will be capitalized, e.g., “Track”, and should be interpreted with their general meaning if not capitalized.

4 DEFINITIONS AND ACRONYMS

4.1 Acronyms

AAC	Advanced Audio Codec
ABNF	Augmented Backus–Naur form
AES	Advanced Encryption Standard
API	Application Programming Interface
AVC	Advanced Video Coding
BMFF	Base Media File Format
CBC	Cypher Block Chaining
CDM	Content Decryption Module
CEA	Consumer Electronics Association
CENC	MPEG Common ENCrytion
CICP	Codec Independent Code Point
CMAF	MPEG Common Media Application Format
CPU	Central Processing Unit
CSS	Cascading Style Sheets
CTA	Consumer Technology Assocation
CTR	CounTeR block cipher mode
DASH	Dynamic Adaptive Streaming over HTTP
DPC	Device Playback Capabilities
DRM	Digital Rights Management
ECMASCRIPT	European Computer Manufacturers Association SCRIPTing
EME	Encrypted Media Extensions
HbbTV	Hybrid Broadband Broadcast Broadband TV
HDCP	High-bandwidth Digital Content Protection
HDMI	High- Definition Multimedia Interface
HDR	High Dynamic Range
HEVC	High Efficiency Video Coding
HLS	HTTP Live Streaming
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IEC	International Electrotechnical Commission

ISO	International Organization for Standardization
ISOBMFF	ISO Base Media File Format
JSON	JavaScript Object Notation
KID	Key IDentifier
LBR	Low BitRate
LSB	Least Significant Bit
MIME	Multipurpose Internet Mail Extensions
MPD	Media Presentation Description
MSB	Most Significant Bit
MSE	Media Source Extensions
NAL	Network Abstraction Layer
PSSH	Protection System Specific Header
SAP	Stream Access Point
SDR	Standard Dynamic Range
SPS	Sequence Parameter Set
TTML	Timed Text Markup Language
UHD	Ultra High Definition
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
VPS	Video Parameter Set
WAVE	Web Application Video Ecosystem
XHTML	Extensible HyperText Markup Language

4.2 Notations

- To access a specific field of an entry in an array, the dot notation is used. For instance, if the *playout* array contains multiple tuples of the form {k=track number,f=fragment number} the notation is as follows:
    - Track of the first entry: *playout[1].k*
    - Fragment number of the first entry: *playout[1].f*
- The numbering of all arrays starts at 1.
- Additional textual descriptions of specific notations are depicted in italic and parentheses, for instance: tf[playout[1,1],1] *(Earliest presentation time of first switching set, first track, first fragment).*

4.3 Abbreviations

`k`	CMAF Track
`tf[k,i]`	Earliest presentation time of CMAF fragment i in CMAF track k
`CF[k,i]`	CMAF Fragment i in CMAF track k

`CH[k]`	CMAF Header of track k
`S[k,f,i]`	Sample number i of track k and fragment f
`Ti`	Measured time when playback is initiated
`TR[s]`	The measured time when sample s is rendered.
`df[k,i]`	The duration of CMAF fragment i in CMAF track k
`CC[k,i,j]`	CMAF chunk j in CMAF fragment i in CMAF track k
`dc[k,i,j]`	Duration of CMAF chunk j in CMAF fragment i in CMAF track k
`br[i]`	The range object i of the corresponding `SourceBuffer`

5 ARCHITECTURE AND WAVE DEVICE REFERENCE MODEL

5.1 WAVE Architecture

The WAVE architecture considers three primary domains: WAVE content, an application based on WAVE APIs, and the WAVE Device Platform. WAVE specifications are an enabler to support:

- Generation of content independently of an application that can be played back on WAVE devices using well defined content formats, playback APIs and device functionalities.
- Implementation of device platforms that enable playback of commonly generated content using well-defined APIs.
- Development of media applications that enable playback of commonly generated content on a broad variety of devices using common APIs.

This architecture enables an ecosystem of independent content generation, app development and device implementations and permits the use of same content within different apps as well as across many different device platforms.

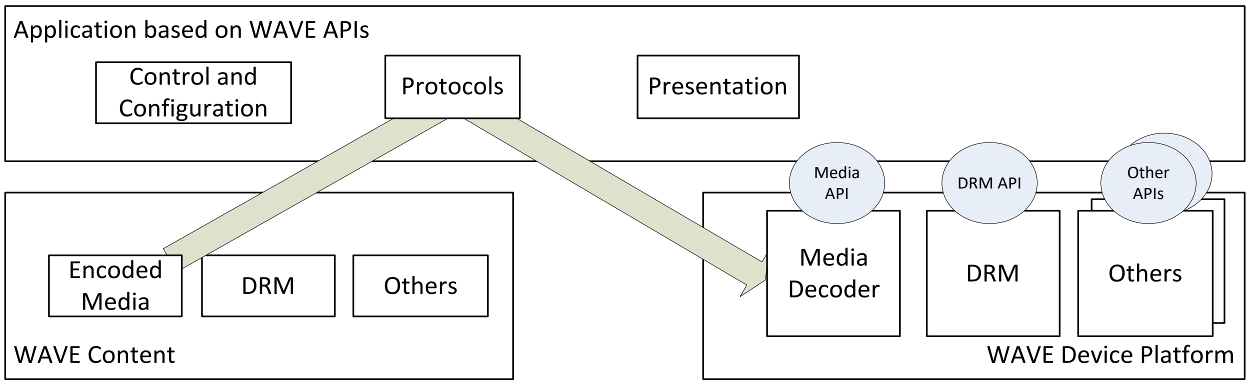


Figure 1: WAVE architecture model

This WAVE specification primarily deals with the requirements of a WAVE Device Platform that may be used by an application implementing WAVE APIs to play back WAVE content. The requirements are purposely held abstract in order to support different application and device interface models. Nevertheless, the use of HTML5 APIs defined in

[WAVE-WMA] is one of the prime objectives. The APIs also differentiate between devices supporting different playback variants, primarily Type 1 and Type 3:

- Type 1 Playback: The WAVE device platform receives a manifest and downloads and plays back the contained media based on the information in the manifest. An application may control the playback with limited control features.
- Type 3 Playback: The WAVE application receives a manifest, downloads the media and uses media APIs in order to playback individual tracks of the media experience. The application is in control of the download and playback of the media using source and track buffers.

5.2 WAVE Device Playback Reference Model

5.2.1 Overview

Based on the architecture in clause 5.1, this clause defines abstracted device models. A device model for Type 3 playback is shown in Figure 3 and for Type 1 playback is shown in Figure 4.

The focus of this initial specification is on Type 3 playback – i.e., the application processes a streaming manifest and controls and schedules the downloading a of CTA WAVE content resources.

This specification follows the HTML-5 and MSE model as shown in Figure 2. A Media Element provides an output and control environment for the playback of media data. A Media Source object represents a source of media data that can be addressed by an application. The *`MediaSource`* combines a list of *`SourceBuffer`* objects that can be used to add media data to the presentation. `MediaSource` objects are created by the application. The application uses the `SourceBuffer` objects to add media data to this source. In addition, Encrypted Media Extensions (EME) and Content Decryption Module (CDM) support APIs and functionalities for decryption.

While this specification abstracts from the concrete instantiation in HTML-5 Media element and MSE instantiation, in case of ambiguities, the terminology for the HTML 5/MSE instantiation applies.

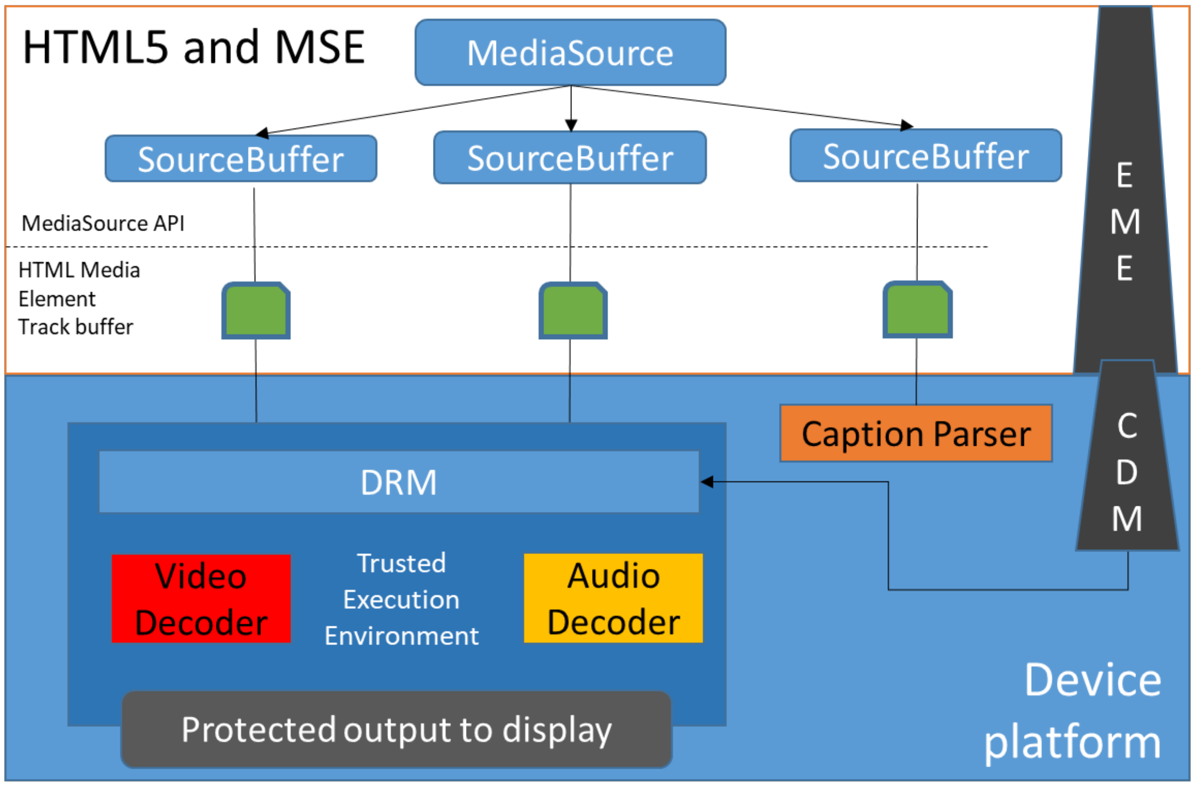


Figure 2: HTML-5 and MSE based Media Source model

In the context of this specification, as CTA WAVE content relies on CMAF content, and CMAF content exclusively uses non-multiplexed tracks, a source buffer is directly mapped to a single-track buffer. Hence, these two terms are used synonymously and interchangeably in the remainder of the specification.

An application that receives a manifest referring to CTA WAVE content is expected to have access to two primary high-level APIs:

- Control API: This API is primarily responsible for capability discovery of the device platform, establishing a Media Element, adding and tearing down media source objects and source/track buffers for specific media types, as well as to control the playback of media.
- Media API: This API consists of one or multiple track/source buffers where the source buffers can be dynamically established and removed. The track/source buffers enable playback of WAVE content by the device platform.

In the case of a streaming application, it is the application dealing with manifest updates as well as with providing and downloading the segments that are then forwarded to the device platform for playback using the media APIs.

The device platform is expected to provide rendering capabilities. Typically, at least a *video display region* and an *audio device* are available as output. The final control of the playback of the media may be handled by the application or it may be part of the device platform. However, it is assumed that a device-oriented mode exists for which playback is primarily the task of the device.

Assuming a device-oriented mode, the device's video and audio output are used to observe if the device platform is capable of fulfilling a set of playback requirements. The application is out of scope for this specification but is assumed that any application that supports the methods and functions of the control and media API can use the device to playback content.

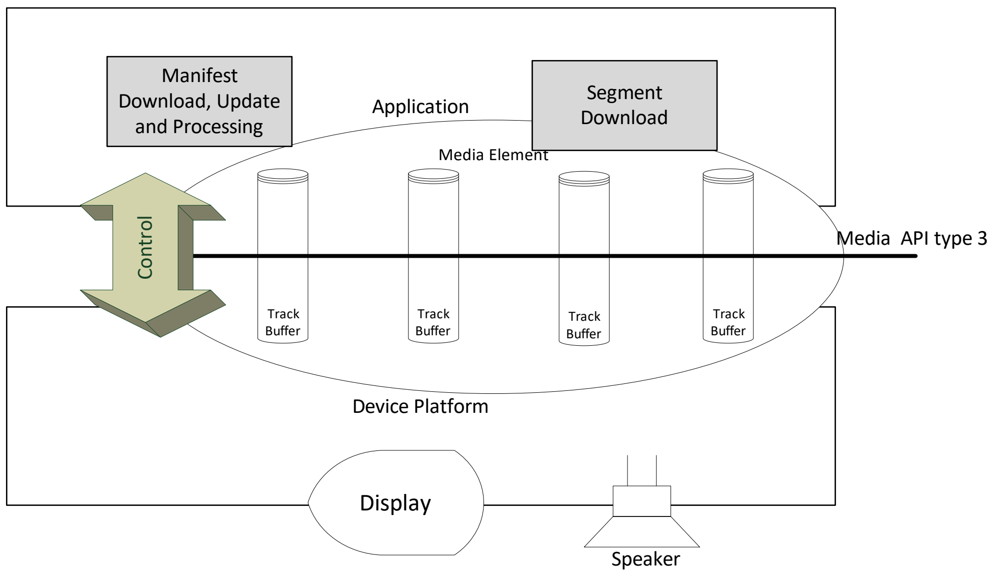


Figure 3: Abstracted device model for Type 3 playback

For type 1 playback, the application has access to the URL of a streaming manifest, but the URL is provided to the device platform through a different type of API. The downloading and processing of the manifest as well as the downloading of the segments is native to the device platform. Communication between application to device playback platform is again split into two primary APIs:

- The control API: This API is primarily responsible for capability discovery of the device platform, establishing and tearing down video elements buffers for specific codecs, as well as to control the playback of media through video elements.
- The media API: this API consists of one or multiple video elements where the video elements can be dynamically established and removed. Each video element enables playback of WAVE content made available through a proper manifest.

Any interfaces between the device platform and external displays or speakers is out of scope of this specification.

In this specification, the primary focus is on type 3 based playback – i.e., it is assumed that the application does manifest download and processing and can establish and playback WAVE content using a media source with an attached set of track buffers.

The application of the requirements in this specification to type 1 playback may be addressed in a future version of the specification.

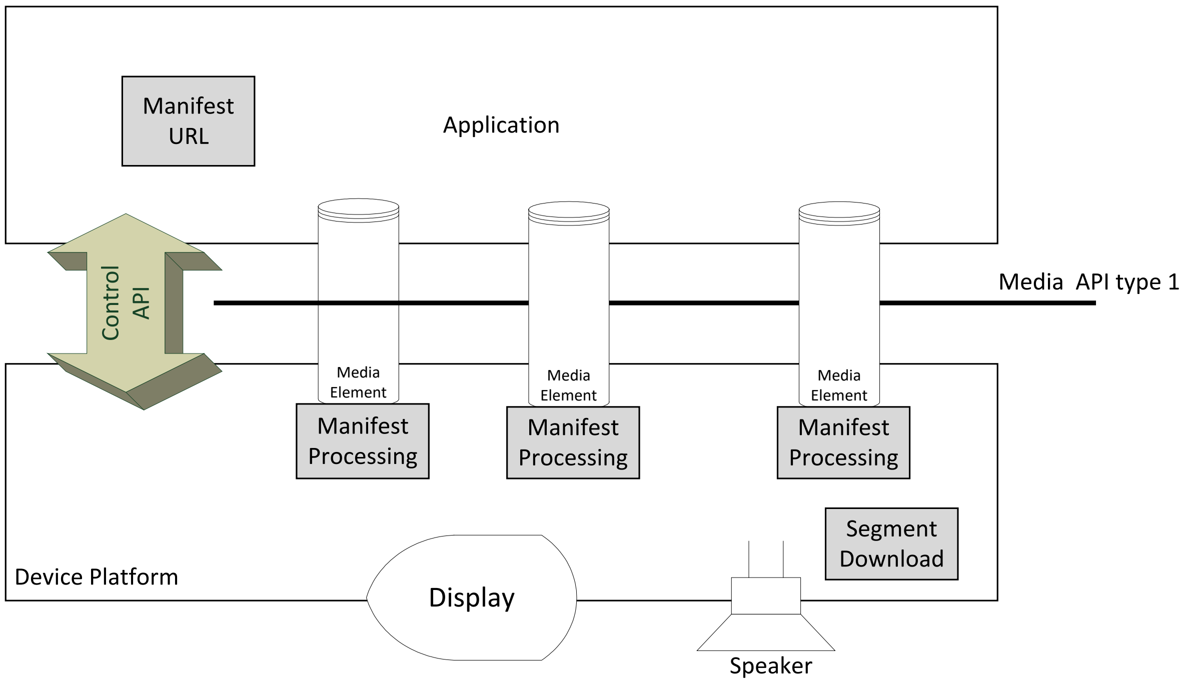


Figure 4: Abstracted device model for Type 1 playback

5.2.2 Wave Device Platform APIs

Following Figure 3, the usage of APIs enables abstraction to the device and playback model. The differentiation between control APIs and media APIs relates to the two complementary WAVE specifications. The control API focuses on the application functions, whereas the media API focuses on playback of WAVE content.

The control API includes all functions to establish, maintain and control media playback. Control APIs are typically used for the initialization phase or at the change of configurations.

The media API includes all functions to append and playback WAVE content through source buffers. With setting up a source buffer for a video media type, either:

- A pre-determined display video output window that matches the aspect ratio and size of the content (height and width) is established, or
- A full-screen display is established.

With setting up a source buffer for an audio media type, an audio output is established.

5.2.3 Web Media API-based Playback Model

This specification is primarily targeted to support Web Media API-based Playback Model [WAVE-WMA].

The following two core specifications are mandated in [WAVE-WMA]:

- HTML 5.1 [HTML51], devices acting as Web browsers that support the HTML syntax and the XHTML syntax, scripting, and the suggested default rendering.
- ECMAScript Language Specification, Edition 5.1 [ECMASCRIPT-5.1].

For Media playback, the following three specifications are mandated in [WAVE-WMA]:

- Encrypted Media Extensions [W3C EME].
- Media Source Extensions [MEDIA-SOURCE].
- Web Audio API [WEBAUDIO] with some exceptions.

It is expected that any device playback tests testing the requirements in this specification relies on the APIs mentioned above.

This specification provides an overview of the key functions and methods that are available for applications, both for testing and for playback. The specification is written in a way such that the functions and methods provided by the above specifications can directly be used for testing and playback.

5.3 WAVE Content

5.3.1 Overview

The WAVE Content [WAVE-CON] specification defines WAVE content. As indicated in clause 3 of [WAVE-CON], WAVE content is based on CMAF [CMAF]. This document also introduces models on the CMAF content model.

WAVE Content is offered as WAVE Programs. A WAVE Program is a sequence of CMAF Presentations, played back sequentially. Sequential playback means that the timing of the playback is controlled by the application, including the ability to create smooth playback without interruptions.

Clause 5.3.2 introduces the CMAF Content Model and clause 5.3.3 introduces the WAVE Content Model based on CMAF.

5.3.2 CMAF Content Model

5.3.2.1 Overview

The CMAF content model is shown in Figure 5.

This specification uses CMAF defined terms according to the CMAF definitions. The key terms the reader of this specification should be familiar with are:

- CMAF Presentation
- CMAF Switching Set
- CMAF Track
- CMAF Header, CMAF Chunk, CMAF Fragment, CMAF Segment
- Decode times of samples
- Presentation times of samples

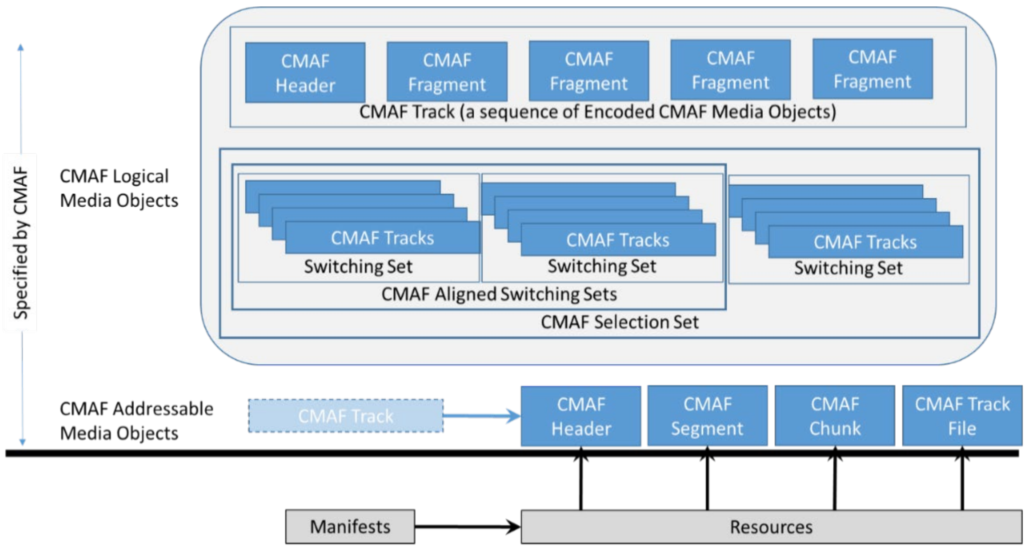


Figure 5: CMAF Content Model

A media sample is media data in a CMAF track associated with a single decode start time and duration.

5.3.2.2 CMAF Addressable Objects

This section provides an overview on CMAF addressable objects, namely:

- CMAF Track Structure in [CMAF].
- CMAF Chunk Structure in [CMAF].
- CMAF Fragment Structure in [CMAF].
- CMAF Segment Structure in [CMAF].

For details, please refer to the CMAF specification [CMAF].

In the remainder of this specification, it is assumed that for test purposes the application can access CMAF Headers, CMAF Fragments, continuous byte ranges of a single CMAF Fragment of arbitrary size, and CMAF Chunks as units and hand those to the media API. The access may for example be through HTTP or any other protocol but could also access the data from a local file storage.

Note that in practical applications, CMAF Fragments and CMAF Chunks may be embedded in Segments (e.g., DASH or HLS) which for itself are the addressable units – i.e., units with an assigned URL. However, for the purpose of testing, it is assumed that CMAF Fragments are addressable (e.g., by a 1-to-1 mapping of CMAF Fragments to Segments) and CMAF chunks are accessible are addressable (e.g., by a 1-to-1 mapping to a DASH delivery unit or by using a chunk index as part of Segment).

5.3.2.3 CMAF presentation timing model

There are multiple timelines involved in playout and rendering CMAF tracks within a presentation.

Each track is a sequence of timed samples. Each sample has a decode time and may also have a composition (display) time offset. Edit lists may be used to over-ride the implicit direct mapping of the media timeline, into the timeline of the overall movie. The movie timeline is used to synchronize CMAF Tracks in a CMAF presentation and also serves as the synchronization source for playback in an HTML-5 media element and the media source.

In addition, each CMAF Track may have assigned an anchor wall-clock time – e.g., UTC time. The wall clock time may be used to relate the relative presentation time of the track to a wall-clock time, for example expressing the time when the corresponding sample was captured, encoded, or packaged.

In summary, three timelines exist and the signaling of the timeline in CMAF is summarized as follows:

- Decode time: The decode time of each CMAF chunk is provided as the `baseMediaDecodeTime` in a `TrackFragmentBaseMediaDecodeTimeBox`. This provides the decode time of the first sample in the CMAF chunk and the remaining decode times are derived by the sample durations in the `'traf'` box.
- Presentation time: The presentation time of each sample in a fragment is determined by the decode time of the sample and, if present, the composition offset (in the sample table) and the track edit list (in the track header). The earliest presentation time in a CMAF Fragment is important for synchronization and switching. Note that the earliest presentation time of a CMAF Fragment may not be the presentation time of the first sample of the CMAF Fragment. In the remainder of the document, presentation time is also referred to as media time.
- Wall-clock time: By the use of a `ProducerReferenceTimeBox` (`'prtf'`), the sample with a specific decode time can be mapped to wall-clock time.

5.3.2.4 CMAF Track Model for this Specification

For the specification, the following definitions are relevant for CMAF Track. For each CMAF Track *k (k=1,…,K)* in a CMAF Switching Set the following features are defined:

- CMAF Header `CH[k], k=1,…,K`
- CMAF Fragments `CF[k,i], i = 1,2,3,… N`
    - Position in the CMAF track `i`.
    - Earliest presentation time: `tf[k,i]`.
    - CMAF Fragment duration: `df[k,i] = tf[k,i+1]-tf[k,i]`.
    - Wall-clock time assigned to the earliest presentation time of CMAF Fragment: `twc[k,i]`.
    - CMAF Chunks `CC[k,i,j], j = 1,2,3,…, C[i]`.
        - Position in the fragment `j`
        - Earliest decode time `tc[k,i,j]`
        - Chunk duration in decode times `dc[k,i,j]`
- An edit list `EL[k]` that may be present in the CMAF header describing the difference between the composition time and the presentation for this track in the CMAF Presentation.
- The earliest presentation time of the first fragment, i.e., `tf[k,i=1]` (presentation time offset).
- The duration of the CMAF Track is defined as `td[k]`.
- The CMAF Track has a assigned a media profile, which includes:
    - CMAF media profile brand.
    - Suitable MIME Type string providing:
        - Media type.
        - Codecs parameter.
        - Profiles parameter <reference>.

- The CMAF Track has samples `sample[k,s]` with `s=1, …, S`, each with nominal presentation time `T[k,s]`.

5.3.2.5 CMAF Switching Set Model for this Specification

The following defines a Switching Set:

- A set of CMAF Tracks conforming to the conditions in clause 5.3.2.4.
- The CMAF Switching Set may contain is single CMAF Header for all CMAF Tracks or an individual CMAF Header for each CMAF Track.
- A Principal CMAF Header CH*. Either the single header or a Principal CMAF Header assigned to the Switching Set `CH*`.

From the definition of a CMAF Switching Set [CMAF], the following holds:

- All CMAF Tracks in a Switching Set conform to one media profile.
- There exists one CMAF Header that is used to initialize the playback of the Switching Set. This header referred as CMAF Principal Header `CH*`.
- The CMAF Header for each track in a Switching Set is defined such that when appending it to the source buffer, it does not result in a reinitialization of the decoding and rendering platform.
- Each CMAF Track in a Switching Set has the same number of CMAF Fragments.
- The earliest decoding time of each CMAF Fragment at the same position `i` in different CMAF Tracks of a CMAF Switching Set are identical.
- The earliest presentation time of each CMAF Fragment at the same position `i` in different CMAF Tracks of a CMAF Switching Set are identical.
- The fragment duration of each CMAF Fragment at the same position in different CMAF Tracks or a CMAF Switching Set are identical.

Note that the equalities above only hold for CMAF Fragments, not necessarily for CMAF Chunks.

5.3.3 WAVE Content Model

5.3.3.1 Overview

WAVE defines a program model and continuous switching sets.

5.3.3.2 WAVE Presentations and Programs

WAVE Programs are a sequence of WAVE Presentations that are played consecutively [WAVE-CON][CMAF]. WAVE presentations in one WAVE program may follow certain constraints and restrictions.

A WAVE Presentation is defined as a collection of WAVE Selection Sets. For each Selection Set, it is expected that one Source Buffer is established using the associated CMAF Principal header and media type. Typically, at least one Selection Set for audio and one Selection Set for video is available.

Each Selection Set contains one or multiple Switching Sets as defined in clause 5.3.2.5.

All tracks in on representation follow the presentation timing model of clause 5.3.2.3.

5.3.3.3 WAVE Splice Constraints

WAVE Splice constraints [WAVE-CON] are defined to ensure seamless playback across presentation boundaries.

5.3.3.4 WAVE Continuous Switching Sets

Continuous switching sets permit to continue playout media without timeline updates.

- For each track *k* in a switching set s that is a continuation of a switching set r the following features are defined and requirements hold.
    - For CMAF Header `CH[s,k] = CH[r,k]`.
    - Decode time of first fragment of s equals decoding time of last fragment plus fragment duration.

The earliest presentation time of the first fragment of the switching set `s` equals to the earliest presentation time of the last fragment of `r` plus the last fragment duration of `r`:

```
tf[s, k, i=1] =  tf[r, k, i=N] + df[s, k, i=N]
```

5.3.3.5 WAVE Splice Conditioned Switching Sets

From the definition of a CMAF WAVE Splice conditioned Switching Sets [WAVE-CON], [CMAF], the following holds for two switching sets `r` and `s`:

- All CMAF Tracks in both Switching Sets conform to one media profile.
- There exists one CMAF Header that is used to initialize the playback of both Switching Sets. This header is referred to as the Principal CMAF Header. Note that this Principal header may be the Principal header of one of the two Switching Sets, or may be superset of the two.
- The CMAF Header for each track in each Switching Set is such that appending it to the source buffer does not result in a reinitialization of the decoding and rendering platform.

5.3.4 Test Content Format

5.3.4.1 General

In order annotate and use CMAF test content, this specification defines a content model format for properly referencing CMAF and WAVE test content. This content model format uses a DASH Media Presentation Description (MPD) [DASH] and is aligned with the DASH core profile for CMAF content as defined in ISO/IEC 23009-1 [DASH], clause 8.12.

Content provided in the context of this specification may be annotated with a DASH MPD.

Content shall only be provided as CMAF Tracks, and not as Track Files.

5.3.4.2 Content Model Format for Single Media Profile

In this clause, test content for a single media profile is defined.

The test content is only valid if the media profile is defined in the CTA WAVE Content Specification [WAVE-CON], clause 4. The media profile is uniquely identified by its media profile identifier `<media profile>` referred to as *CMAF Brand* in the tables in the WAVE Content Specification. Examples are `'cfhd'` or `'caac'`.

Content that includes more than one media profile is addressed in clause 5.3.4.3. However, in general the combination of content can be addressed by the test runner, and hence the focus in this specification is the preparation for appropriate test content for a single media profile.

CMAF Tracks in CTA WAVE content may be provided in two different modes

- Fragment Mode: Every CMAF Fragment is mapped to a single DASH Segment.

- Chunk Mode: Every CMAF Chunk is mapped to a single DASH Segment.

In order to annotate CTA WAVE test content appropriately, a DASH MPD is used, and the test content shall be provided as follows:

1. The DASH Media Presentation shall conform to the DASH Core Profile for CMAF content as defined in ISO/IEC 23009-1 [DASH], clause 8.12.5.
2. On MPD level, the following is provided:
    - At least two `@profiles` shall be present, namely `urn:cta:wave:test-content-media-profile:2021` as well as `urn:mpeg:dash:profile:cmaf:2019`. Setting `@profiles` to `urn:cta:wave:test-content-media-profile:2022` means that the MPD follows the requirements in this clause.
    - The `ProgramInformation` element shall be present with the following information:
        - The `Title` element should be present as an instructive summary of the Test Asset. Examples are:
            - `Croatia, 1920 x 1080, 60fps, cfhd, Test Vector 1`
            - `ToS, 1280 x 720, 30fps, cfhd, Test Vector 2`
        - The `Source` element shall be present be set to "`CTA WAVE - <first mezzanine> version <version from json> (<creation date from json>)`".
            - Example: "`tos_L1_1920x1080@30_60 version 2 (2021-08-04)`".
        - The `Copyright` element shall be present and shall bet set to:
            - For ToS video: "`© Tears of Steel (https://mango.blender.org/about/) (2012) by Blender Institute, credited to Ton Roosendaal and Ian Hubert, used and licensed under Creative Commons Attribution 3.0 Unported (CC BY 3.0) (https://creativecommons.org/licenses/by/3.0/)by the Consumer Technology Association (CTA)® / annotated, encoded and compressed from original.`"
            - For Croatia video: "`© Croatia (2019), credited to EBU, used and licensed under Creative Commons Attribution 4.0 International (CC BY 4.0) (https://creativecommons.org/licenses/by/4.0/) by the Consumer Technology Association (CTA)® / annotated, encoded and compressed from original.`"
            - For white noise audio: "`© Consumer Technology Association (CTA)®, licensed under Creative Commons Attribution 4.0 International (CC BY 4.0) (https://creativecommons.org/licenses/by/4.0/) / annotated, encoded and compressed from original.`"
        - The `@moreInformationURL` should be present and provide a relative reference to an additional document providing more information on this Test Asset including source, contact, production workflow as well as a report from a validator that shows that the content is valid. More details on exact information is for further study.
    - `@type` shall be set to `static`.
    - `@mediaPresentationDuration` shall be present and is set to the presentation duration of the contained Switching set `td`.

- Only a single **`Period`** element shall be present.

3. On Period level, the following is provided:
    - Exactly one Adaptation Set shall be provided.
4. For this Adaptation Set, the following is provided:
    - The `@containerProfiles` parameter shall be present and shall include at least two profile strings, namely:
        - A structural brand being either `'cmfc'` or `'cmf2'`. Note that both profiles may be signalled in case the content conforms to `'cmf2'`.
        - The CMAF media profile brand as defined in the tables in the WAVE Content Specification.
    - A CMAF Principal Header should be identified using `@initializationPrincipal`. If not present, any CMAF Header of the Adaptation Set may be used for capability checks and initialization.
    - The `@codecs` shall be set to the according to the sample entry `codingname` field of the CMAF Principal Header (based on CMAF Profile requirement) and following the definition in the CTA WAVE content specification [WAVE-CON] for the media profile and shall be one of the `'codecs'` MIME subparameters as listed in the CTA WAVE content specification [WAVE-CON] for this media profile.
    - If the Content is protected, then `ContentProtection` element shall be present and set according to the requirements for the DASH Profile for CMAF content with the additional constraints as follows:
        - A `ContentProtection` element with `@schemeIdUri` equal to `"urn:mpeg:dash:mp4protection:2011"` and `@value` equal to an encryption scheme defined in ISO/IEC 23001-7 and in the CTA-WAVE Content Specification – i.e., `'cenc'` or `'cbcs'`. It may contain a `@cenc:default_KID` attribute. If present, the value of this attribute shall match the `'tenc'` box `default_KID` value.
        - At least one `ContentProtection` element with `@schemeIdUri` equal to a UUID value that signals that content keys can be obtained with a DRM system identified by the UUID. It may contain a `@cenc:default_KID` attribute and a `cenc:pssh` element. If present, the values shall match the `tenc` box `default_KID` value and the `moov` box `pssh` value respectively.
5. For every Representation, the following is provided:
    - The id shall be set to the name of the used mezzanine excluding any file ending, for example:
        - `"croatia_A1_480x270@12.5_60"`
        - `"tos_I1_1024x576@30_60"`
    - The `SegmentTemplate` element shall be present with the following constraints:
        - The `@startNumber` shall be absent (i.e., assumed to 1).
        - The `@initialization` attribute shall be present and shall be set to:
            - `"$RepresentationID$/init.cmfv"` for video.
            - `"$RepresentationID$/init.cmfa"` for audio.
            - `"$RepresentationID$/init.cmft"` for text.

- The `@media` attribute shall be present and shall be set as follows from MPD to the location of the CMAF Fragment/Chunk with:
    - For Fragment Mode: `"$RepresentationID$/$Time$.m4s"` with CMAF Fragments `CF[k,i], i = 1,2,3,… N` are made available at `"$RepresentationID$/tf[k,i].m4s"`.
    - For Chunk Mode: `"$RepresentationID$/$Time$_$SubNumber$.m4s"` with CMAF Chunks `CC[k,i,j], j = 1,2,3,…, C[i]` are made available at `"tf[k,i]_<j>.m4s"`.
- The `SegmentTimeline` element shall be present.

6. If
    1. Chunking is consistent within one Adaptation Set, and
    2. Timescale is the same on each CMAF tracks in an Adaptation Set,

   then `SegmentTemplate` as defined above should be used on Adaptation Set level.

   Note 1: The above format is not built for efficiency, as it is only considered for content exchange.

   Note 2: A content model for providing one media profile in two or more aligned Switching Sets is still to be defined.

Example MPD for a Fragmented Switching Set

```
<?xml version="1.0" ?>
<!-- MPD file Generated with GPAC version 2.1-DEV-rev420-g9f8725974-master at 2022-10-
17T09:39:06.106Z -->
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT2.000S" type="static"
mediaPresentationDuration="PT0H0M30.000S" maxSegmentDuration="PT0H0M2.000S"
profiles="urn:mpeg:dash:profile:isoff-
live:2011,urn:mpeg:dash:profile:cmaf:2019,urn:cta:wave:test-content-media-profile:2022">
  <ProgramInformation>
    <Title>tos, 1920x1080, 30.0fps, cfhd, Test Vector 1</Title>
    <Source>tos_L1_1920x1080@30_30 version 3 (2022-04-20)</Source>
    <Copyright>© Tears of Steel (https://mango.blender.org/about/) (2012) by Blender Institute,
credited to Ton Roosendaal and Ian Hubert, used and licensed under Creative Commons Attribution
3.0 Unported (CC BY 3.0) (https://creativecommons.org/licenses/by/3.0/) by the Consumer
Technology Association (CTA)® / annotated, encoded and compressed from original.</Copyright>
  </ProgramInformation>
  <Period duration="PT0H0M30.000S">
    <AssetIdentifier schemeIdUri="urn:cta:org:wave-test-mezzanine:unique-id" value="0.1"/>
    <AdaptationSet segmentAlignment="true" maxWidth="1920" maxHeight="1080" maxFrameRate="30"
par="16:9" lang="und" startWithSAP="1" contentType="video" containerProfiles="'cmf2',
'cfhd'">
      <SegmentTemplate media="1/$Time$.m4s" initialization="1/init.mp4" timescale="15360">
        <SegmentTimeline>
          <S t="0" d="30720" r="14"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="1" mimeType="video/mp4" codecs="avc1.640028" width="1920"
height="1080" frameRate="30" sar="1:1" bandwidth="5100000">
      </Representation>
    </AdaptationSet>
  </Period>

</MPD>
```

Example MPD for a Chunked Switching Set with Content Protection

```
<?xml version="1.0" ?>
<!-- MPD file Generated with GPAC version 2.1-DEV-rev420-g9f8725974-master at 2022-10-
17T09:39:06.106Z -->
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT2.000S" type="static"
mediaPresentationDuration="PT0H0M30.000S" maxSegmentDuration="PT0H0M2.000S"
profiles="urn:mpeg:dash:profile:isoff-
live:2011,urn:mpeg:dash:profile:cmaf:2019,urn:cta:wave:test-content-media-profile:2022">
  <ProgramInformation>
    <Title>tos, 1920x1080, 30.0fps, cfhd, Test Vector 1</Title>
    <Source>tos_L1_1920x1080@30_30 version 3 (2022-04-20)</Source>
    <Copyright>© Tears of Steel (https://mango.blender.org/about/) (2012) by Blender Institute,
credited to Ton Roosendaal and Ian Hubert, used and licensed under Creative Commons Attribution
3.0 Unported (CC BY 3.0) (https://creativecommons.org/licenses/by/3.0/) by the Consumer
Technology Association (CTA)® / annotated, encoded and compressed from original.</Copyright>
  </ProgramInformation>
  <Period duration="PT0H0M30.000S">
    <AssetIdentifier schemeIdUri="urn:cta:org:wave-test-mezzanine:unique-id" value="0.1"/>
    <AdaptationSet segmentAlignment="true" maxWidth="1920" maxHeight="1080" maxFrameRate="30"
par="16:9" lang="und" startWithSAP="1" contentType="video" containerProfiles="'cmf2',
'cfhd'">
      <SegmentTemplate media="1/$Time$.m4s" initialization="1/init.mp4" timescale="15360">
        <SegmentTimeline>
          <S t="0" d="30720" k="4" r="14"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="1" mimeType="video/mp4" codecs="avc1.640028" width="1920"
height="1080" frameRate="30" sar="1:1" bandwidth="5100000">
     </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

5.3.4.3 Content Model Format for Multiple Media Profiles

A content model for multiple media profiles in one MPD is for further study.

5.4 Detailed Scope of This Specification

5.4.1 Introduction

This specification deals with providing requirements following the description in Figure 6. Devices with capabilities defined in this specification based on playback can play back WAVE content using well defined APIs following the model from above. Playback includes different aspects, and for each of these aspects the APIs and the units of WAVE content are used in different ways to stimulate the playback.

If a device supports certain capabilities (that can possibly by queried by an external application), then using well-defined API calls as well as WAVE content when stimulating media track buffers enables playback of media content.

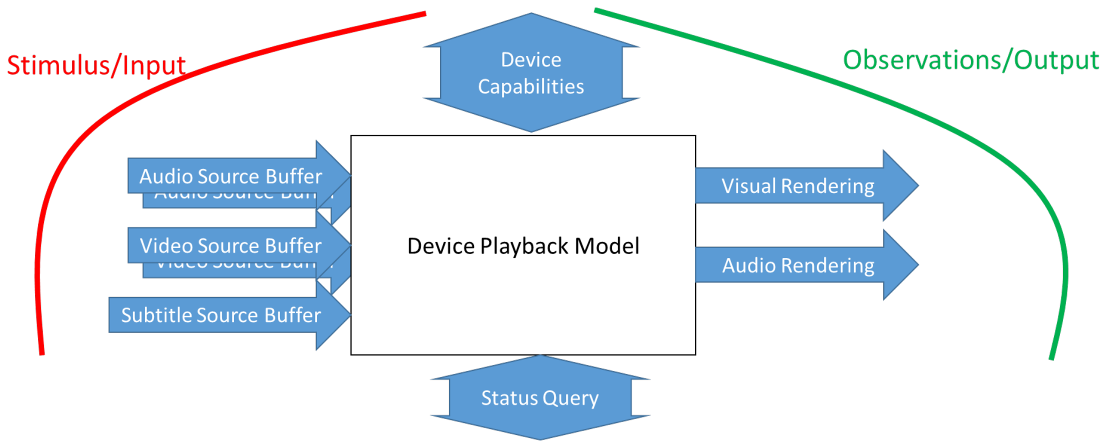


Figure 6: Device playback model

5.4.2 Conformance Aspects and Interoperability

This specification defines different conformance and interoperability points for devices:

- **Media Profile Playback**: Enables *playback* of a CMAF Media Profile as part of a WAVE Presentation.
- **Device Core Profiles**: Enable *playback* of WAVE Programs conforming to a ***WAVE Program "Profile".***
  - A full set of media profile playback capabilities that permit playback of WAVE Presentations.
  - Capability of transition across different WAVE Presentations.
  - Additional requirements such as media synchronization, etc.
- **Device Extension Profiles**: Enable *playback* of enhanced experiences beyond a Device Core Profile if the content provides the extension and the device supports the extensions.
  - Examples are media profiles, or for example seamless switching across codecs, etc.
- **Configurations:** Enable *playback* of a Device Core Profile or Device Extension profile with a specific configuration by supporting at least one configuration.

The initial conformance points provided in this specification is media profile playback conformance.

It is expected that in later versions of the specification, the primary conformance points for this specification will be the device core profiles. Device core profiles collect a set of requirements, and a device conforms to a device profile if it conforms to all requirements that are associated to that device profile. A device core profile supports a full audio-visual experience to play back a WAVE Presentation Profile.

5.4.3 Tests, Performance and Performance Requirements

This specification is written such that the tests can be defined. Tests describe the usage of playback APIs and the expected resulting observations. Such observations may be documented at high-level (observation-based), but it is preferable that well-defined and measurable performance requirements are defined. Whereas an ideal performance may be desirable and should be documented, in certain cases a degradation of the ideal performance may also be acceptable be sufficient. In this case not only the ideal performance is documented, but also minimum performance requirements for a device.

This specification is supported by a test suite; for details refer to clause 5.5.

5.4.4 Existing and New Devices

This specification is written primarily for the development and testing of new devices. Device platform manufacturers are expected to take into account the requirements when developing new devices. In this case the device maker should target to meet the device playback performance requirements as documented in this specification.

However, this specification also permits testing existing devices for their performance and to document the performance and suitability for WAVE content playback. In this case, even if a device does not fulfill the full performance requirements, the device may still be used for playback as long as some minimum performance requirements are fulfilled.

This specification can be used for both cases.

5.5 CTA WAVE Device Capability Test Suite

5.5.1 Overview

In order to support this specification, an extensive test suite is defined and provided by CTA WAVE. The test suite includes well-defined audio-visual test content for many different parameters, including different codecs, and different content parameters such as resolutions, frame rates, encrypted content and more. The content is used by a test environment to test the core functions of a device for media playback including linear playback, seamless switching, random access, A/V synchronization, handling non-linear/trick play, and splicing. The content is properly annotated to permit both manual and automatic testing to ensure that the test under device performs correctly.

Figure 7 provides an overview of the CTA WAVE Device Playback Test Suite.

- **Original Content Pool**: The original content used for the test suite includes widely available open-source content such as "Tears of Steel" and "Big Buck Bunny" as well as a 50Hz stream specially filmed in Croatia by the European Broadcasting Union (EBU) for testing purposes.
- **Content Annotation**: This module implements scripts to generate annotated test content based of raw source content that is compatible with the DPC test suite. For details refer to Annex C.
- **Mezzanine Content Pool**: This is a pool of annotated content that meets the requirements of the Device Playback Capabilities Tests.
- **Media Profile Encoding**: This module builds test content generated from the Mezzanine Content using different codecs and varying media profiles.
- **Content Verifier**: This module implements a test to check whether content conforms to CTA WAVE and CMAF content.
- **MPD + Test Content**: This module hosts CTA WAVE test content that conforms to CTA WAVE content (as defined in [WAVE-CON]) as well as the test content format defined in clause 5.3.4 and can be used in the test execution framework to test content playback.
- **DPC Tests (Test Runner Instructions)**: The DPC Tests comprise HTML and JavaScript code together with configuration information for code that is re-used between tests. For more details about the configuration information, see references to "tests.json" in this document.
- **Test Content Usage Instructions**: Part of the configuration information is structure called “playout” which defines that defines which CMAF fragments from which tracks are to be played out in what order.

- **Test Runner**: The DPC Test Runner implements an execution environment for tests following the WAVE DPCTF Test Specification. Test Runner uses generated CTA test content and curated test cases.
- **Device Observation Framework**: The DPC Device Observation Framework determines compliance with this specification, based on observations including video recordings of tests which are run on a device by the Test Runner.
- **Media API**: The media API includes all functions to append and playback WAVE content through source buffers.
- **Control API**: The control API includes all functions to establish, maintain and control media playback. For further details on the Media and Control APIs refer to Sections 5.2.1 and 5.2.2.

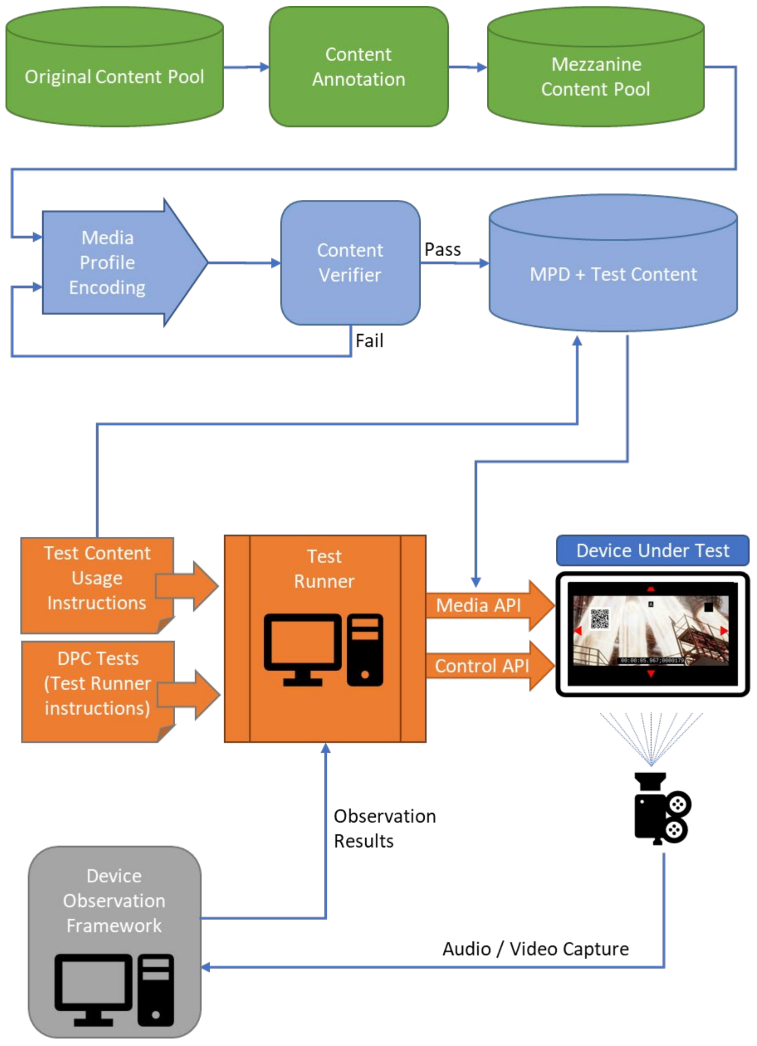


Figure 7: CTA WAVE Device Playback Test Suite Overview

5.5.2 Resources and Links

An overview of resources associated to the WAVE Device Playback Test Suite are as follows:

DPC Landing Page	https://cta-wave.github.io/dpc-test-suite

Generating mezzanine content	https://github.com/cta-wave/mezzanine
Generating test content	https://github.com/cta-wave/Test-Content-Generation
Test content	https://cta-wave.github.io/Test-Content https://github.com/cta-wave/Test-Content
Content verifiers	https://conformance.dashif.org https://github.com/cta-wave/test-content-validation
DPC Deploy	https://github.com/cta-wave/dpctf-deploy
Individual tests	https://github.com/cta-wave/dpctf-tests
Test runner	https://github.com/cta-wave/dpctf-test-runner
Observation Framework	https://github.com/cta-wave/device-observation-framework

6 MEDIA PLAYBACK MODEL

6.1 Introduction

For an application to make use of the device platform for playback of CTA WAVE content, it requires the following principal actions:

1. Establishing a *Media Element* that enables the control and presentation of media data.
2. Create a Media Source that serves as the source of media data for a *Media element.*
3. Attach a *Media Source* to the *Media Element*.
4. Checking if the device is *capable* of adding a Source Buffer for playback of CMAF content in general, as well as the media profile contained in a CMAF Switching Set.
5. If supported, attach a *Source Buffer* initialized with the appropriate parameters that enables the playback of the WAVE content through a well-defined set of APIs.
6. Append CMAF Headers as an Initialization Segment to *Source Buffers.*
7. Append CMAF Addressable Resources (Segments, Chunks) as a Media Segment.
8. Apply methods to Media source and individual Source Buffers to support different functionalities.
9. Start and control playback of the *Media Element.*

A device should only establish a Source Buffer for a media type if the device is capable of playing back at least one of the included WAVE media profiles for each media type in the WAVE content offerings.

Once a source buffer for a media type is established, this source buffer is used for the playback of the media type.

6.2 Media Element Establishment and Source Buffer Attachment

6.2.1 General

A *Media Element* presents an output and control environment for WAVE content playback.

The *Media Source* represents a source of WAVE content that is consumed by a *Media Element*.

Source Buffers are attached for specific media playback. All *Source Buffers* attached to one *Media Element* share a common media time. A *Media Element* is established for playback of a WAVE Presentation and a *Media Source* is attached to it to dynamically add WAVE content.

The Media Element permits playback control functions such as play, pause, seek, etc.

6.2.2 Web Media API-based Media Element and Source Establishment

The video element as defined in HTML-5 is a media element whose media data is ostensibly video data, possibly with associated audio data. The `src`, `preload`, `autoplay`, `loop`, `muted`, and `controls` attributes are the attributes common to all media elements. For details see https://www.w3.org/TR/html51/semantics-embedded-content.html#the-video-element.

The following attributes are of relevance:

`width:` attribute specifies the width of a video player, in pixels.

`height:` attribute specifies the height of a video player, in pixels.

`style = "width:` <horizontal dimension>`px; height:` <vertical dimension>`px".`

Note that it is preferred to use the style attribute instead of the direct width and height attributes. The width and height attributes are leftovers from HTML before the arrival of CSS and the concept of the separation between content and style.

The Media Source Extensions [MEDIA-SOURCE] Specification defines the *Media Source* object. The *Media Source* object represents a source of media data for an HTMLMediaElement. It also includes a list of `SourceBuffer` objects to add media data to the presentation. `MediaSource` objects are created by the web application and then attached to an HTMLMediaElement. For details, see https://www.w3.org/TR/media-source/#MediaSource.

```
// 1. Establishing a Media Element that enables the control and presentation of
media data
<!DOCTYPE html>
<style>
    video {
        width: <width from MPD>;
        height: <height from MPD>;
    }
</style>
<body>
  <video controls="true"></video>
  <script type="text/javascript">
     // see below
  </script>
```

```
</body>
</html>
// 2.       Create a Media Source that serves as the source of media data for a
Media element
var MediaSource = new MediaSource(); // MediaSource.readyState === 'closed'
// 3.       Attach a Media Source to the Media Element
// Get video element
var video = document.querySelector('video');
// Attach media source to video element
video.src = URL.createObjectURL(MediaSource);
// Wait for media source to be open
MediaSource.addEventListener('sourceopen', handleSourceOpen.bind(MediaSource));
```

6.3 Media Element and Media Source Control

6.3.1 General

The Media Element can be controlled and monitored for the playback. For this, several methods are supported for playback, primarily the `play()` method. This method initiates playback. Another method is `pause()`. Beyond this, the Media Element playback rate can be adjusted by using a `playbackRate()` function, and a `seek()` function can be applied by setting the playback position to a specific time.

Furthermore, some properties of the media element can be monitored, namely:

- The time ranges of the `buffered` media.
- The `duration` of the media resource, in seconds.
- The `currentTime` current time of the playback position.

The Media element provides among others at least the following events with the condition when the event is fired:

- `ended`: Playback has stopped because the end of the media was reached.
- `loadeddata`: The first frame of the media has finished loading.
- `pause`: Playback has been paused.
- `play`: Playback has begun.
- `seeked`: A seek operation completed.
- `seeking`: A seek operation began.
- `timeupdate`: The time indicated by the `currentTime` attribute has been updated.
- `waiting`: Playback has stopped because of a temporary lack of data.

6.3.2 Web Media API-based Media Element and Media Source Control

In the Web Media API [WAVE-WMA] context, the above properties and methods are available. The mapping of the generic properties from clause 6.3.1 to Web Media API [WAVE-WMA] context is provided in Table 1.

Table 1 Mapping of the generic properties to Web Media API Context

General function	Mapping to Web Media API
`play()`	`HTMLMediaElement.play()` as defined in https://www.w3.org/TR/html51/semantics-embedded-content.html#playing-the-media-resource
`pause()`	`HTMLMediaElement.pause()` as defined in https://www.w3.org/TR/html51/semantics-embedded-content.html#playing-the-media-resource
`seek()`	Set `currentTime` as defined in https://www.w3.org/TR/html51/semantics-embedded-content.html#offsets-into-the-media-resource
`currentTime`	Get `currentTime` as defined in https://www.w3.org/TR/html51/semantics-embedded-content.html#offsets-into-the-media-resource.
`duration`	`duration` as defined in https://www.w3.org/TR/html51/semantics-embedded-content.html#offsets-into-the-media-resource
`buffered`	`buffered` as defined in https://www.w3.org/TR/html51/semantics-embedded-content.html#loading-the-media-resource

All events are summarized here: https://www.w3.org/TR/html51/semantics-embedded-content.html#mediaevents.

For convenience and completeness of this spec, this is documented in Annex B.0.

6.4 Device Capability

6.4.1 General

WAVE content may be offered with different options such that an application can choose based on its device capabilities. If none of the options are supported, the device cannot playback the content. The WAVE content specification defines multiple media profiles for each media type. Hence, an application should determine the device capabilities and based on this select the proper content options. Device capability discovery is an essential feature for an Internet media-based ecosystem.

In order to discover whether content playback is possible, basically three aspects need to be checked:

1) Support of CMAF content format.
2) Support of a specific CMAF media profile.
3) Support of content protection requirements, if the content is encrypted.

Aspect #3 is addressed in clause 7 on the playback of encrypted content.

CMAF is a restricted subset of the ISO BMFF Byte Stream format: https://www.w3.org/TR/mse-byte-stream-format-isobmff/ [ISO-BMFF-BYTE-STREAM]. For details refer to Annex C.

On the support for CMAF content, the query for a the support of the ISO BMFF byte stream format as defined in https://www.w3.org/TR/mse-byte-stream-format-isobmff/ [ISO-BMFF-BYTE-STREAM], and the relevant parameters are defined through the MIME-types for this specification using `"audio/mp4"` and `"video/mp4"` as defined in RFC 6381. In addition, for the support of CMAF, the structural brands `'cmfc'` and `'cmf2'` as defined in ISO/IEC 23000-19 matter. Hence, to test the capability for CMAF playback can be done using a query according including a MIME Type with sub-parameter profiles as follows:

- `audio/mp4 profiles='cmfc'` or `audio/mp4 profiles='cmf2'`
- `video/mp4 profiles='cmfc'` or `video/mp4 profiles='cmf2'`

In order to test for the specific CMAF media profile, several approaches may be used.

The first approach using the MIME type parameter in an equivalent approach as for the structural brand. In this case, two options exist:

1) A support query for the media profiles is carried out with argument `video/mp4 profiles="<brand>"` and results in a `yes` with `<brand>` specific for each profile.
2) A support query for the media profiles is carried out with argument `video/mp4 codecs="<codecs>"` results in a `yes` as defined in the WAVE Content Specification [WAVE-CON], Table 1.

In an extended approach more detailed media capabilities can be queried. In this case not only the codecs parameter is used, but also additional functions may also be queried. Details on the use of media capabilities is documented in Annex A. The mapping of media profiles to media capability API calls is discussed with each media type and possibly each media profile separately.

In yet another approach the application queries the device if the content described in the CMAF header can be played back. This has the advantages of being complete, accurate, future-proof, but the drawback of not being human readable and possibly requires transmitting more information than the other approaches. In addition, there may be cases that not all information is sufficiently provided in the CMAF Header, for example in case with Inband Parameter Sets.

For all queries, the device may respond with:

- `Yes`: If yes is provided, then the playback requirements for this media profile as documented in this specification are expected to be fulfilled.
- `No`: If no is provided, then the playback of the media profile is not supported by the device and the application shall not playback this media profile.
- `unknown`: In this case the application should find other options to identify if the media profile can be played back.

Based on the implementation variety, this specification does not mandate a specific device capability discovery. It is recommended that an application uses all four methods to identify whether support of a media profile is possible.

For each media profile, a detailed capability discovery mechanism is provided in this specification.

If none of the above methods are applicable or sufficiently conclusive, it is recommended that a source/track buffer with the media type is established as defined in clause 6.5, and the successful establishment is used as an indication for successful playback.

6.4.2 Web Media API-based Capability Discovery

If a media source is to be created, the `MediaSource.isTypeSupported(type)` method may be used.

- For details, see: https://www.w3.org/TR/media-source/#dom-MediaSource-istypesupported.
- The method is expected to return `"false"` or `"true"`.

In the context of this specification, the MIME-types for this specification must conform to the rules outlined for `"audio/mp4"` and `"video/mp4"` in RFC 6381 – i.e., the ISO BMFF Byte Stream format is used as defined in [MSE-FORMAT-ISOBMFF].

The proper `type` for each media profile is defined in the WAVE content specification [WAVE-CON]. If the device responds with to this method with true, the device claims to support playback of a specific WAVE media profile, then this specification defines the requirements that need to be fulfilled.

As an alternative to the `isSupportedType` method, a CMAF Header may be appended once the Source Buffer is established (see clause 6.5 for details) and then the source buffer monitored for any error.

Media Capabilities API [MEDIACAPABILITIES] provides an improved alternative to the `isTypeSupported()` API together with the MIME Type parameters introduced clause 6.4.1 for determining whether a given user agent is capable of encoding, decoding and rendering a piece of content. To determine if a user-agent can decode a particular piece of content, the `mediaCapabilities.decodingInfo()` method is called. The method takes an instance of a `MediaDecodingConfiguration` object as an argument and returns as a Promise a `MediaCapabilitiesInfo` object.

For details refer to Annex A and the individual media profiles.

One example option is provided below.

```
// 4. Checking if the device is capable of adding a Source Buffer for playback
of CMAF content
// in general, as well as the media profile contained in a CMAF Switching Set.
var mimeCodec = 'video/mp4; profiles="cmfc,cfhd" codecs="avc1.4D401F"';
console.log(MediaSource.isTypeSupported(mimeCodec));
```

6.5 Source Buffer Management

6.5.1 General

In order to make use of the platform for media playback, the source buffer needs to be added and initialized properly.

A WAVE device shall support an API such that the application can create a new source buffer for a specific media type and adds the buffer to the media source buffer list.

A WAVE-compliant device shall allow the establishment of at least one source buffer for media type `video` and one for media type `audio`. A WAVE device should support the establishment of a source buffer for the playback of `subtitles`. A WAVE device may support the establishment of multiple source buffers for each media type.

A WAVE device shall support an API to establish a source buffer for CMAF content playback.

A WAVE device shall support an API such that the application can remove a source buffer from the media source buffer list.

For all media types, a WAVE device shall support an API such that the application can update the configuration of the source buffer. It is recommended that the source buffer is only updated if the update follows the requirements documented in the remainder of this specification. Otherwise, it is recommended to remove the source buffer and re-establish a new one with the new configuration.

A source buffer is expected to be established for each media type individually by:

1) Adding a source buffer providing the media type, possibly augmented with sub-MIME parameters.
2) If 1 is successful, initialize the source buffer with an appropriate CMAF Principal header.
3) Create a proper output environment for each established source buffer.
    a. For video a pre-determined display window that matches the aspect ratio and either
        i. default to the size of the content (height and width) of the CMAF Principal Header is established,
        ii. or as an option a full screen mode may be used as well.
    b. For audio, no specific provisioning is done unless the media profile defines a specific output environment.

In addition, successful establishment of the source buffer also implies that the set of media APIs documented in clause 6.6 are supported and can be used by the application for media playback.

6.5.2 Web Media API-based Source Buffer Management

The Media Specification defines the `MediaSource.addSourceBuffer(type)` method.

- For details, see https://www.w3.org/TR/media-source/#dom-MediaSource-addSourceBuffer.
- The method returns a source buffer.

The `type` shall follow the requirements in ISO BMFF Byte Stream Format [MSE-FORMAT-ISOBMFF], clause 2.

The source buffer is further initialized by appending a CMAF header (CH) to the `SourceBuffer` by using the `MediaSource.appendBuffer(CH)`. For a CMAF Switching Set, the CMAF Principal Header shall be used.

The source buffer may be updated/re-initialized by appending a CMAF header (CH) to the `SourceBuffer` by using the `MediaSource.appendBuffer(CH)`.

The source buffer is further managed by the `MediaSource.removeSourceBuffer(type)` method.

- For details, see https://www.w3.org/TR/media-source/#dom-MediaSource-removeSourceBuffer.
- The method does not return any value.

The display window is established by using width and height from the CMAF Header.

```
// 5. If supported, attach a Source Buffer initialized with the appropriate
// parameters that enables the playback of the WAVE through a well-defined
// set of APIs.
var SourceBuffer = MediaSource.addSourceBuffer(mimeCodec);
```

6.6 Device Playback Model for a Single Source Buffer

6.6.1 Introduction

Once a media element, a media source and a source buffer are established, media playback can be established using a set of well-defined APIs and functions. This specification does not provide requirements for the functional availability of these media APIs but assumes that these functions are available. For instance, these media APIs can be realized with W3C Media Source Extensions [MEDIA-SOURCE] as listed in clause 6.2.2.

Furthermore, it is assumed that:

- A source buffer is established with some configuration parameters according to the requirements in clause 6.5, and
- The capability discovery in clause 6.4 resulted in an acknowledgement that the announced WAVE content can be played back.

Functions and properties described in the following are:

- Methods and parameters to support source buffer operation and playback.
- Observations that permit to query the state of the source buffer and the playback.

Such functions and methods may be general or may be specific for specific media types. A simplified playback model for a single media type is provided inFigure 8.

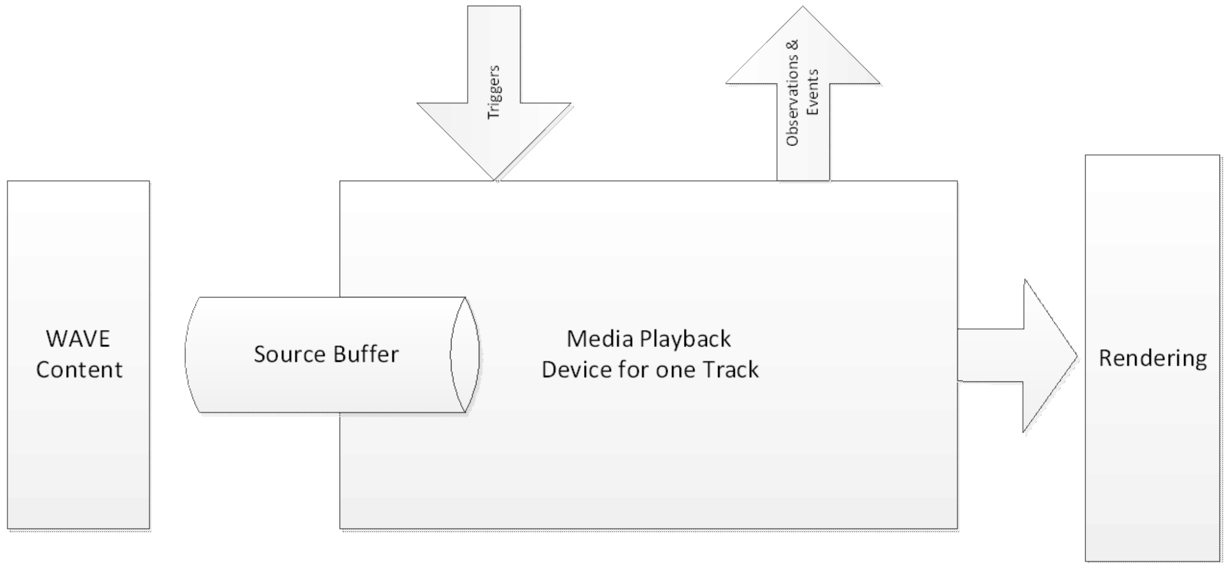


Figure 8: Simplified playback model

6.6.2 General

This clause outlines the source buffer model for this specification. It describes the various rules and behaviors associated with appending data to a `SourceBuffer`.

At the highest level:

- An application creates `SourceBuffer` objects and appends sequences of CMAF data to update their state.
- The media element pulls media data out of the `MediaSource` object, plays it, and fires events.
- The application is expected to monitor these events, for example to determine when it needs to append more CMAF media data.

The `SourceBuffer` is expected to basically implement algorithms aligned to those defined in [MEDIA-SOURCE], clause 3.5. It also inherits the methods from the Media Element or Media Source as defined in clause 6.7.

The key method is the `appendBuffer` functionality. In the context of CTA WAVE specification, this method has the following parameters:

- `data`: the appended data is a CTA WAVE addressable resource. the appended data is added to the buffer by including the data in the timeline and possibly overwriting existing data in the buffer.
- `offset`: optional parameter to signal the offset that needs to be applied to the presentation time in order to map it to the media timeline. If not provided the parameter is assumed to be 0.
- `appendWindow`: optional parameter providing a presentation timestamp range used to filter out coded frames while appending. The append window represents a single continuous time range with a single start time and end time. Coded frames with presentation timestamp within this range are allowed to be appended to the `SourceBuffer` while coded frames outside this range are filtered out.

Once this method is invoked, but the buffer is unable to accept data, the device is expected to run proper methods, for example:

- Append Error Algorithm as defined in [MEDIA-SOURCE], clause 3.5.3
- Coded Frame Eviction Algorithm as defined in [MEDIA-SOURCE], clause 3.5.10

If invoked and the buffer is able to accept data, the underlying platform is expected to run the following steps to process the appended data (note that this is a high-level description on what is defined in [MSE]):

- The appended data `data` is parsed.
    - If invalid data not conforming to CTA WAVE content is found, an appropriate error algorithm is run, and an event is fired. The appended data `data` is ignored and not appended.
    - Remove any bytes that the byte stream format specifications require to be ignored from the start of the input buffer. The byte stream specification format for CMAF data follows the MP4 ISO BMFF Byte Stream Format [MSE-FORMAT-ISOBMFF].
    - If the parsed data `data` is a CMAF Header:
        - Run the CMAF Header Append received function.
    - If the parsed data `data` is a CMAF Chunk or CMAF Fragment:
        - Apply the offset.
        - Add the data to the buffer.
    - If the parsed data `data` is any other part of a CMAF Fragment:
        - Apply the offset.
        - Add the data to the buffer.

The Header append function:

- If the appended data does not conform to the CMAF Header, ignore the `data` and stop.
- If the appended CMAF Header has a non-recognized media profile, ignore, fire an event and stop.
- If the appended CMAF Header is not the first CMAF Header that is appended and processed, check the following:
    - Is the same media type received? If yes, does the codecs and all other parameters announced in the updated CMAF Header match was specified in the CMAF Principal Header?
    - If not, create an event that informs application about an incompatible header. The application may run a Change Type function as documented below.
- If all successful, add track descriptions to track buffers:
    - Track description is byte stream specific.
- Set the need for a random-access point for the media.

The Change Type function:

- The Change Type function allows to change the codec by setting the MIME type differently for future calls for the append functions. This makes it possible to change codecs or container type mid-stream. After executing this function, a Header append is expected.

Media Append function:

- If the Source Buffer is not initialized with a CMAF Header (should never be the case as the `SourceBuffer` Establishment adds header), then ignore the data.
- If the appended data `data` contains one or more coded frames/samples, then add each sample within the `appendWindow` to the track buffer as follows:
    - Let `dts`, `pts` and `dur` be the decoding time stamp, presentation time stamp and duration of the sample.
    - Adjust to `pts` and to `dts` with the offset value.
    - If random access is needed and sample is no random access point, drop frame.
    - If an overlapped frame is detected:
        - Run overlap algorithm, depending on media type (see below).
    - Remove existing coded frames from track buffer with non-matching presentation time.
    - Remove all frames that depend on any frames removed in the previous two steps.
    - Add coded frame/sample with `pts`, `dts` and `dur` to track buffer.
    - Set highest time stamp for track and also set group end timestamp.
- Extend the duration of the track buffer.
- The media append function allows to *set the timestamp offset* to control the offset applied to timestamps inside media segments that are subsequently appended to the `SourceBuffer`.

Overlap algorithm:

- If media is appended, then:

- For video, blending is applied, i.e., the new media starts at the earliest presentation time of the new appended media. The samples of the previous media are overwritten.
- For audio, refer to MSE
  - https://www.w3.org/TR/media-source-2/#SourceBuffer-audio-splice-frame-algorithm
  - https://www.w3.org/TR/media-source-2/#SourceBuffer-audio-splice-rendering-algorithm
- For text, refer to MSE:
  - https://www.w3.org/TR/media-source-2/#SourceBuffer-text-splice-frame-algorithm

6.6.3 Methods

Assuming the definitions from above, once the capability for the playback of a CMAF Switching Set is identified, the following functionalities are be assumed for playback platform (not all of them are necessarily supported):

Initialization: When adding a CMAF Principal Header for a CMAF Switching Set, then the media playback platform is initialized using the Header append as defined in clause 6.6.2. Should the playback platform not be able to playback the media in this CMAF Track, it would throw an Error. *(`SourceBuffer`)*

Add Media Data: When adding a CMAF Media Segment (Fragment, Chunk), then the media is processed accordingly and added to the buffer using a media append algorithm as defined in clause 6.6.2. Should the appended data not be able suitable, it would throw an Error. Otherwise the source buffer is updated with the media. *(`SourceBuffer`)*

Playback Start: After adding a first media segment/CMAF Fragment to the media playback pipeline with an earliest presentation time `t0`, and the playback is started at wall-clock time `Tstart`, then the media presentation time `t0` is anchored to wall-clock time `Tstart`. The playback is in playback state with state variables `(t0, T0=Tstart)`. *(MediaElement)*

Playback Pause: A playback pause at a wall-clock time `Tpause` pauses the presented media at media time `tp = (Tpause - Tstart) + t0`. The media buffer is in pause state with variable `tp`. *(MediaElement)*

Playback Resume: If the media playback is in pause state at time `tp`, resuming the playback at time Tresume, so anchors the presentation time `t` is anchored to wall-clock time `(t - tp) + Tresume`. *(MediaElement)*

Playback Teardown: At a wall-clock time `Tteardown`, the playback (in playback or pause state) of the media may be stopped at the associated media time. *(MediaElement)*

Playback over Gaps: If in playback state with state variables `(t0, T0)` and no media is available for wall-clock time `T` with presentation time `t = (T - T0) + t0`, the media timeline virtually advances (for example by stalling the presentation) until a CMAF Fragment is received with earliest presentation time `tept` that is then presented at `Tept = (t - t0) + T0`. *(MediaElement and Application Logic)*

Time adjustment: In between the appending of data and in playback state with `(t0, T0)`, the media playback may be time-adjusted with a value `toffset` in a sense that the playback of all media appended after the adjustment, the presentation time is offset by this constant value. Hence, after this offset is applied, media samples with presentation time `t` are presented at wall-clock time `(t - t0 + toffset) + Tstart`. The state of the playback changes to `(t0+toffset, T0)`. *(Source Buffer)*

Overlaps: If data from the start of a new CMAF Fragment is appended, for which the presentation time is already included in the media pipeline (also possibly right after a time adjustment has happened), then the later appended media data overwrites the earlier included media data on the media presentation timeline. *(`SourceBuffer`)*

Removal: Data in the source buffer may be removed from a `start` position towards and `end` position. *(`SourceBuffer`)*

Truncation: If data from the start of a new CMAF Fragment is appended, the content to be played back may be truncated at the beginning of the CMAF Fragment or at the end of the CMAF Fragment or both by using an *append window start* or *append window end* signal that specifies the presentation times of the first and last sample to be presented. If not set, the entire CMAF Fragment is played back. *(`SourceBuffer`)*

Switching: The insertion of a CMAF header of a different CMAF Track in the same CMAF Switching Set after the insertion of a CMAF Fragment at position *i* and then inserting the CMAF Fragment of the new CMAF Track at position *i*+1. Such switching is considered seamless. In certain circumstances, switching from track to another may be done by not initializing with a CMAF Header, if the CMAF Header is the same or at least equivalent for each CMAF Track in the CMAF Switching Set (referred to as *Bitstream Switching* in contrast to the generic *Header Switching*). *(`SourceBuffer`)*

ChangeCodec: At a specific media time `ti`, the source buffer is changed (typically with a new codecs parameter and different CMAF Principal Header) and the playback is started such that the earliest presentation time `t0` of this new media is mapped to the wall-clock time `(t - t0) + Tstart.` Typically, change codec is expected to be seamless. *(`SourceBuffer`)*

Re-Initialization: At a specific media time `ti`, the *media source* is *teardowned*, newly *initialized* and adding new source buffers (typically with a different codecs and CMAF Principal Header at least for one of the source buffers) and the playback is started such that the earliest presentation time `t0` of this new media is mapped to the wall-clock time `(t - t0) + Tstart.` Typically, such a re-initialization is not seamless. *(`SourceBuffer`)*

6.6.4 Web Media API-based Playback

6.6.4.1 Introduction

This clause documents the methods and functions of the media source.

The mapping of the functions defined in clause 6.6.3 to a Web Media API-based playback is provided.

6.6.4.2 Assumptions

It is assumed that an MPD according to the test format defined in clause 5.3.4.2 is available. Elements and attributes are referenced accordingly.

Before executing any of the below, it is assumed that at least the following steps have been successfully completed:

- 6.2.2: Web Media API-based Media Element and Source Establishment

- 6.4.2: Web Media API-based Capability Discovery
- 6.5.2: Web Media API-based Source Buffer Management

```
// 1. Establishing a Media Element that enables the control and presentation of media data
<!DOCTYPE html>
<style>
    video {
        width: <AdaptationSet@width from MPD>;
        height: <AdaptationSet@height from MPD>;
    }
</style>
<body>
  <video controls="true"></video>
  <script type="text/javascript">
    // see below
  </script>
</body>
</html>

// 2. Create a Media Source that serves as the source of media data for a Media element
var MediaSource = new MediaSource(); // MediaSource.readyState === 'closed'

// 3. Attach a Media Source to the Media Element
// Get video element
var video = document.querySelector('video');
// Attach media source to video element
video.src = URL.createObjectURL(MediaSource);
// Wait for media source to be open
MediaSource.addEventListener('sourceopen', handleSourceOpen.bind(MediaSource));

// 4. Checking if the device is capable of adding a Source Buffer for playback of CMAF
// content in general, as well as the media profile contained in a CMAF Switching Set.
var mimeCodec = '<AdaptationSet@mimeType>; profiles="<AdaptationSet@containerProfiles>"
codecs="<AdaptationSet@codecs>"';
console.log(MediaSource.isTypeSupported(mimeCodec));

// 5. If supported, attach a Source Buffer initialized with the appropriate parameters
// that enables the playback of the WAVE through a well-defined set of APIs.
var SourceBuffer = MediaSource.addSourceBuffer(mimeCodec);
```

Additional assumptions are provided for each function.

6.6.4.3 Initialization

Basic Function: When adding a CMAF Principal Header for a CMAF Switching Set, then the media playback platform is initialized using the Header append as defined in clause 6.6.2. Should the playback platform not be able to playback the media in this CMAF Track, it would throw an Error. *(`SourceBuffer`)*

Assumption:

- The assumptions from clause 6.6.4.2.

- A CMAF Principal Header is available as header, either through **`AdaptationSet`**`@initializationPrincipal` or, if not present, by using any header of the Adaptation Set provided by **`AdaptationSet`**.**`SegmentTemplate`**`@initialization.`

Process:

- `SourceBuffer.appendBuffer(header);`

Post-Condition:

The source buffer is initialized unless an error is reported. If an error is reported, playback of the switching set is not supported.

6.6.4.4 Add Next Media data

Basic Function: When adding a CMAF Media Segment (Fragment, Chunk), then the media is processed accordingly and added to the buffer using a media append algorithm as defined in clause 6.6.2. In order to add the correct media data in a timeline, the start time of the media segment is extending the buffered range. Should the appended data not be able suitable, it would throw an Error. Otherwise, the source buffer is updated with the media. (*`SourceBuffer`*)

Assumption:

- The assumptions from clause 6.6.4.2.
- The initialization from clause 6.6.4.3.
- The source Buffer is initialized with the CMAF Header corresponding to this CMAF Track.
- A CMAF Media Segment is available as data by appending a media segment provided through `AdaptationSet.SegmentTemplate@media`.

Process:

- `SourceBuffer.appendBuffer(data);`

Post-Condition:

The source buffer is extended with data of the time range covered in the CMAF media segment.

6.6.4.5 Playback Start

Basic Function: After adding a first media segment/CMAF Fragment to the media playback pipeline with an earliest presentation time `t0`, and the playback is started at wall-clock time `Tstart`, then the media presentation time `t0` is anchored to wall-clock time `Tstart`. The playback is in playback state with state variables `(t0, T0=Tstart)`. *(MediaElement)*

Assumption:

- The post-conditions of 6.6.4.4 apply – i.e., the source buffer includes media data.

Process:

- `HTMLMediaElement.play()`

Post-Condition:

The media plays and the playback position `currentTime` advances with real-time.

6.6.4.6 Playback Pause:

Basic Function: A playback pause at a wall-clock time `Tpause` pauses the presented media at media time `tp = (Tpause - Tstart) + t0`. The media buffer is in pause state with variable `tp`. *(MediaElement)*

Assumption:

- The post-conditions of 6.6.4.5 apply – i.e., the data in the source buffer is played.

Process:

- `HTMLMediaElement.pause()`

Post-Condition:

The media playback pauses, and the playback position remains constant at `currentTime`.

6.6.4.7 Playback Resume:

Basic Function: If the media playback is in pause state at time `tp`, resuming the playback at time `Tresume`, the presentation time `t` is anchored to wall-clock time `(t - tp) + Tresume`. *(MediaElement)*

Assumption:

- The post-conditions of 6.6.4.6 apply – i.e., the playback element is paused at `currentTime`.

Process:

- `HTMLMediaElement.play()`

Post-Condition:

The media plays and the playback position `currentTime` advances with real-time.

6.6.4.8 Playback Teardown:

Basic Function: At a wall-clock time `Tteardown`, the playback (in playback or pause state) of the media may be stopped at the associated media time. *(MediaElement)*

Assumption:

The assumptions from clause 6.6.4.2 – i.e., a `SourceBuffer` is created.

Process:

- `HTMLMediaElement.removeAttribute('SourceBuffer');`
- `HTMLMediaElement.load();`

Post-Condition:

The source buffer is removed.

6.6.4.9 Playback over Gaps

Basic Function: If in playback state with state variables `(t0, T0)` and no media is available for wall-clock time `T` with presentation time `t = (T - T0) + t0`, the media timeline virtually advances (for example by stalling the presentation) until a CMAF Fragment is received with earliest presentation time `tept` that is then presented at `Tept = (t - t0) + T0`. *(MediaElement and Application Logic)*

Assumption:

A time range is available in the buffer that is not continuous

Process:

- The media element in itself does not support a function to address this. However, an application logic performs the principle as follows such that `seekToPosition` is set to the start of first range after the gap. Once this seek is completed, playback is resumed. The logic also needs to take into account the duration of the gap to when resume playback.
- An example implementation is provided here `https://github.com/Dash-Industry-Forum/dash.js/blob/development/src/streaming/controllers/GapController.js` `(298 - 306)`

Post-Condition:

After the seek, playback is continuous.

6.6.4.10 Time adjustment

Basic Function: In between the appending of data and in playback state with `(t0, T0)`, the media playback may be time-adjusted with a value `toffset` in a sense that the playback of all media appended after the adjustment, the presentation time is offset by this constant value. Hence, after this offset is applied, media samples with presentation time `t` are presented at wall-clock time `(t - t0 + toffset) + Tstart`. The state of the playback changes to `(t0+toffset, T0)`. *(Source Buffer)*

Assumption:

A value `tOffset` is available.

Process:

- `SourceBuffer.timestampOffset = tOffset;`

Post-Condition:

Any media appended after this offset is set is played back with the offset.

6.6.4.11 Overlaps:

Basic Function: If data from the start of a new CMAF Fragment is appended, for which the presentation time is already included in the media pipeline (also possibly right after a time adjustment has happened), then the later appended media data overwrites the earlier included media data on the media presentation timeline. *(`SourceBuffer`)*

Assumption:

- A buffer is already present that includes media that is not yet presented spanning some time ranges.
- A maximum time threshold is set for the duration of the overlap media. A typical value is the fragment duration.
- A media fragment data is available with a presentation time that includes a presentation times that are at least partially already in the buffer, but the smallest media time is larger than the sum of `currentTime` and the media fragment duration.

Process:

- `SourceBuffer.appendBuffer(data)`

Post-Condition:

The source buffer has an updated time range that spans the appended media data and replaces media data that was previously in the buffer with new media data.

6.6.4.12 Remove

Basic Function: Remove data from buffer within time window [start, end]. *(`SourceBuffer`)*

Assumption:

The source buffer includes media from time range start to end.

Process:

- `SourceBuffer.remove(start, end)`

Post-Condition:

The range is adjusted to remove the data in between start and end.

6.6.4.13 Truncation:

Basic Function: If data from the start of a new CMAF Fragment is appended, the content to be played back may be truncated at the beginning of the CMAF Fragment or at the end of the CMAF Fragment or both by using an *append window start* or *append window end* signal that specifies the presentation times of the first and last sample to be presented. If not set, the entire CMAF Fragment is played back. *(`SourceBuffer`)*

Assumption:

- The assumptions from clause 6.6.4.2.
- The initialization from clause 6.6.4.3.
- The source Buffer is initialized with the CMAF Header corresponding to this CMAF Track.

- An *append window start* or *append window end* value are available that are within the time range of the appended `data`.

Process:

- `buffer.appendWindowStart = appendWindowStart;`
- `buffer.appendWindowEnd = appendWindowEnd;`
- `SourceBuffer.appendBuffer(data);`

Post-Condition:

The source buffer is extended with data of the time range covered in the CMAF media segment and the overlap of the time window.

6.6.4.14 Switching:

Basic Function: The insertion of a CMAF header of a different CMAF Track in the same CMAF Switching Set after the insertion of a CMAF Fragment at position *i* and then inserting the CMAF Fragment of the new CMAF Track at position *i*+1. Such switching is considered seamless. In certain circumstances, switching from one track to another may be done by not initializing with a CMAF Header, if the CMAF Header is the same or at least equivalent for each CMAF Track in the CMAF Switching Set (referred to as *Bitstream Switching* in contrast to the generic *Header Switching*). *(`SourceBuffer`)*

Assumption:

- A CMAF Header is available as header, provided by `AdaptationSet.SegmentTemplate@initialization` for the CMAF Track switched to.
- A CMAF Media Segment is available as data that is the i+1 of the switch to track `AdaptationSet.SegmentTemplate@media`.

Process:

- `If (!bitstreamswitching)`
    - `SourceBuffer.append(header)`
- `SourceBuffer.append(data)`

Post-Condition:

The source buffer is extended with data of the time range covered in the CMAF media segment.

6.6.4.15 Change Codec

Basic Function: At a specific media time `ti`, the source buffer is changed (typically with a new codecs parameter and different CMAF Principal Header) and the playback is started such that the earliest presentation time `t0` of this new media is mapped to the wall-clock time `(t – t0) + Tstart`. Typically, change codec is expected to be seamless. *(`SourceBuffer`)*

Assumption:

- The assumptions from clause 6.6.4.2.
- The initialization from clause 6.6.4.3.
- The source Buffer is initialized with the CMAF Header corresponding to this CMAF Track.
- Media is in the source buffer up to a specific time `ti`.
- New media data `data` is available with start time `ti` for which the codec/mimeType changes.

Process:

- `SourceBuffer.changeType(mimeType)`
- `SourceBuffer.append(header)`
- `SourceBuffer.append(data)`

Post-Condition:

- The source buffer is updated accordingly, and the media is appended for continuous playback.

6.6.4.16 Re-Initialization

Basic Function: At a specific media time `ti`, the *media source* is *torn down*, newly *initialized* and adding new source buffers (typically with a different codecs and CMAF Principal Header at least for one of the source buffers) and the playback is started such that the earliest presentation time `t0` of this new media is mapped to the wall-clock time `(t – t0) + Tstart`. Typically, such a re-initialization is not seamless. *(MediaElement)*

Assumption:

- The assumptions from clause 6.6.4.2.
- The initialization from clause 6.6.4.3.
- The source Buffer is initialized with the CMAF Header corresponding to this CMAF Track.
- Media is in the source buffer up to a specific time `ti`.
- New media data `data` is available with start time `ti` for which the codec/mimeType changes.

Process:

- `MediaSource.removeSourceBuffer(buffer);`
- `element.removeAttribute('src');`
- `element.load()`
- `src = new MediaSource()`
- `video.src = URL.createObjectURL(MediaSource);`

-
  ```
  MediaSource.addEventListener('sourceopen',
  handleSourceOpen.bind(MediaSource));
  ```

-
  ```
  var mimeCodec = '<AdaptationSet@mimeType>;
  profiles="<AdaptationSet@containerProfiles>"
  codecs="<AdaptationSet@codecs>"'
  ```
- `console.log(MediaSource.isTypeSupported(mimeCodec));`

        o `var SourceBuffer = MediaSource.addSourceBuffer(mimeCodec);`
    - `SourceBuffer.changeType(mimeType)`
    - `SourceBuffer.append(header)`
    - `SourceBuffer.append(data)`

Post-Condition:

- The source buffer is updated accordingly, and the media is appended for continuous playback.

6.7 Device Playback Model for a Media Element

6.7.1 General

For each media type contained in the WAVE content, a Source Buffer is established.

Each source buffer then consumes the media of the corresponding type.

6.7.2 Web Media API-based Playback

This clause documents the methods and functions of the media source. The details will be added in a future version of the specification.

NOTE: It is expected that the development of the test environment will support this documentation.

7 DRM PROTECTED MEDIA

7.1 Introduction

Content security is accomplished in a device by two functions:

1. Key management by a digital rights management system (DRM), which authenticates a user or their device and authorizes decryption and playback of a CMAF Track on that device under control of the DRM client and under specific conditions – e.g., output protection, rental period, hardware root of trust, etc.

2. Decryption of a CMAF Track encrypted with Common Encryption using either the `'cenc'` or `'cbcs'` encryption scheme.

Each DRM system has a proprietary trust infrastructure for DRM client private and public key pairs and encrypted media keys. DRM systems normally use a proprietary license format that contains encrypted keys and playback conditions specified in a rights expression language. For example, in the case of FairPlay™, the key server only passes an encrypted key and must rely on server authorization rules (such as checking a valid rental period) and built-in device functionality (such as requiring output protection). These systems cryptographically bind authorization to a specific device and content (with one or more Tracks encrypted with that media key), which prevents a license from being used with unintended content or on unintended devices.

The operation, implementation, and robustness of each DRM system is considered out of scope for CTA WAVE and in particular this specification. DRM systems are assumed to be actively managed and tested to maintain content security by testing client implementation security, identifying and repairing breaches, revoking compromised keys, denying keys to compromised devices, securely storing licenses, maintaining a secure processing environment and memory, integrating with security hardware to protect keys and decrypted media samples, etc. A WAVE device can select one or more DRM systems to implement based on commercial factors such as DRM features, robustness, availability, popularity, provider support, cost, content publisher requirements, etc.

What is in scope for CTA WAVE and this specification is:

- Correct parsing and decryption of CMAF Tracks encrypted with `'cenc'` or `'cbcs'` encryption schemes specified in ISO/IEC 23001-7 Common Encryption and as defined in CTA WAVE content specification [WAVE-CON], and
- The use of Encrypted Media Extensions (EME) APIs to request and download a device-supported DRM license or key to test decryption, decoding, and rendering of protected content.

Different DRM systems support different functions, such as persistent licenses, hierarchical licenses, license rotation, key rotation, multiple keys per license, sample variants, stream counting, secure stop, higher security for UHD content, etc. The primary focus of WAVE testing is the basic functions common to all DRM systems with the assumption that DRM-specific tests can be provided by each DRM system.

Note that the W3C EME specification [W3C EME] also defines “ClearKey” decryption in browsers using Common Encryption. The ClearKey system ensures that there is a common baseline level of functionality that is guaranteed to be supported in all user agents, including those that are entirely open source. Thus, content providers that need only basic decryption can build simple applications that will work on all platforms without needing to work with any content protection providers.

In the context of this release of the specification, only ClearKey-based decryption is tested. Other DRM systems may be added in the future.

7.2 Media Profiles and Encryption Schemes

7.2.1 Introduction

The WAVE Content specification allows Common Encryption using either the `'cenc'` or `'cbcs'` scheme with specific restrictions and constraints defined in the CTA WAVE Content Specification [WAVE-CON], clause 4.5.

The media and encryption formats supported by W3C Encrypted Media Extensions are specified here: https://www.w3.org/TR/eme-stream-registry/ and https://www.w3.org/TR/eme-stream-mp4/.

NOTE: As quoted below, the W3C EME ISOBMFF stream specification only includes the ‘cenc’ scheme and it only references CENC 2nd edition.

Abstract

This specification defines the stream format for using ISO Base Media File Format [ISOBMFF] content that uses the ISO Common Encryption ('cenc') protection scheme [CENC] with the Encrypted Media Extensions [W3C EME].

Note:
Although the ISO Base Media File Format [ISOBMFF] associated with this format's MIME type/subtype strings supports multiple protection schemes, when used with Encrypted Media Extensions, these strings refer specifically to content encrypted and packaged using the 'cenc' protection scheme [CENC].

The 3rd edition of CENC (ISO/IEC 23001-7:2016) added the `'cbcs'` scheme and manifest signaling. Hence the W3C EME ISO BMFF stream format needs to be updated.

To apply the existing specification for the ‘cbcs’ scheme, replace each instance of the 4CC code ‘cenc’ with ‘cbcs’, "CTR mode" with "CBC mode", and reference the 2016 edition of CENC.

7.2.2 Test Runner for encrypted content using EME API

The initData for ISOBMFF is specified as the payload of a ‘pssh’ box and represented in a DASH manifest as an element containing a base64 encoded ‘pssh’ box and an attribute equal to the SystemID of the DRM system.

Although a ‘pssh’ box may be included in a file header for each DRM system supported (useful for downloaded files without a manifest), for streaming it is preferable to signal *initData* in the manifest so it can be processed once and the necessary licenses can be identified and downloaded in advance of attempting to play the encrypted media.

For test content related to this specification, the `pssh` init data is always specified in the DASH MPD. In case the `initData` is also included in the media segments as a `pssh` box, the EME implementation will dispatch an *encrypted* event. The application shall ignore this event and use the init data from the manifest.

1. When a test playlist signals that a test may contain encrypted content, the test runner will query what key systems and configurations are supported using *requestMediaKeySystemAccess(*DOMString *keySystem,* sequence<MediaKeySystemConfiguration> *supportedConfigurations).*

   The sequence of `MediaKeySystemConfiguration` configurations are tried in order. The first element with a satisfiable configuration is used. A configuration is an audio/video content type, optionally including the ‘codecs’ subparameter, and information on the required robustness level.

   The application needs to differentiate between DRM systems. In the context of this specification, only Clearkey is supported. For the details of identifying support for different DRM systems, refer to Annex A.2.

2. The test runner calls *createMediaKeys()* to create a new *MediaKeys* object for *keySystem.* This object has a *sessionId*, collects events, and exposes the *MediaKeyStatusMap*, which lists KIDs known to the session and the status of each key.
3. In case DRM system uses a server certificate, the certificate is set using the MediaKeys createdMediaKeys.setServerCertificate(serverCertificate).
4. The test runner calls *createSession()* to start a *MediaKeySession* that groups all calls, messages, and events for this key. There can be multiple simultaneous sessions – e.g., for different audio and video keys, subsequent presentations, etc.

5. When a license request is signaled, the test runner calls *generateRequest(*DOMString *initDataType* BufferSource *initData)* on the key session so the CDM will format a license request from DRM *initData*, as specified in https://www.w3.org/TR/eme-initdata-registry/ and https://www.w3.org/TR/eme-initdata-cenc/. Once the license payload has been generated by the CDM, the application/player receives the payload via the `"message"` event `keySession.addEventListener("message", licenseRequestReady, false).`
6. The application forwards a CDM-formatted license request to a license server for the selected key system, usually with an access token. A license server must encrypt a license or key with a key bound to the CDM under test, so dynamic generation and download of licenses is required during testing. Note that for Widevine, the license server is not included in the init data. Consequently, player implementations usually provide an API interface that allows applications to specify the license server for the required DRM system.
7. The downloaded license is sent to the CDM by the test runner using session.*update(*`BufferSource` *`response`).*
8. The runner may need to create additional sessions and license requests for additional key IDs used by other audio or video Switching Sets and start playback when all the necessary keys have been updated on the CDM.

NOTE: For details on testing playback of encrypted content, please refer to:

- https://github.com/cta-wave/dpctf-tests/blob/master/lib/player.js
- https://github.com/cta-wave/dpctf-tests/blob/master/playback-of-encrypted-content-https.html

Details on how to use the MPD test content information in the `ContentProtection` element for encrypted playback are for further study.

8 SINGLE-TRACK MEDIA PLAYBACK REQUIREMENTS

8.1 Introduction and Content Model

This clause documents typical functionalities required by an application in the playback using a source model as documented above for a single track.

A single CMAF track is assumed.

If not mentioned otherwise, fragmented content is used and not chunked content. If chunked content is to be used, the test explicitly states this.

8.2 Sequential Track Playback

8.2.1 Background

Sequential track playback refers to a CMAF/WAVE track played from the beginning by providing CMAF fragments to the source buffer after initialization.

8.2.2 Pre-condition

A CMAF Track is available for playback following the properties in clause 5.3.2.4, with the earliest presentation time `tf[k,i=1]=0`. All relevant information for initialization is provided, for example through a test content manifest as defined in 5.3.4.2.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5 including the relevant output. This includes:

1) Appending the CMAF Header for the track.
2) Establishing a proper output environment.

8.2.3 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback. The recommended value is 1 second.
2) `TSMax`: The maximum permitted startup delay set to 120ms.

NOTE: This constraint is defined as a first approach but may be refined after running some initial tests.

8.2.4 Stimulus

For a source buffer that supports a media profile as initialized properly, Sequential Playback of a CMAF Track k, consisting of a sequence of CMAF Header and CMAF Fragments, refers to the following actions:

- *Append* the CMAF Header `CH[k]` to the Source Buffer.
- *Set time* offset to `tf[k,i=1]`.
- *Add media data* using CMAF Fragments `CF[k,i]` starting from `i=1` . Add as many CMAF fragments `CF[k,i]` starting from fragment 1 such that the buffer duration is at least `min_buffer_duration`.
- Once reached, *start play-back* on the media source and observe:
    - o The measured time when playback is initiated is `Ti`.
    - o The measured time when the first sample is rendered at time `TR[k,s=1]`.
    - o The measured time when sample `s` is rendered is time `TR[k,s]`.
- While it is not the last fragment:
    - o As soon as the `buffer` is `min_buffer_duration` or below, apply *add media data* using the next fragment `CF[k,i]`
- Stop at the end of the last CMAF Fragment in the buffer.

8.2.5 Required Observations

8.2.5.1 General

If the above algorithm is carried out, the following observations are expected:

1) The playback duration shall match the duration of the CMAF Track – i.e., `TR[k,S] = TR[k,1] + td[k]` taking into account:
    - Missing starting and ending frames,
    - Potential start frames being rendered before playback,
    - Frozen last frame after the playback ended.
2) The end of the playback shall be announced.

8.2.5.2 Video

If the track is a video track, then the following additional observations are expected:

1) Every video frame S[k,s] shall be rendered such that it fills the entire video output window following the properties in clause 5.2.2.
2) The presented sample shall match the one reported by the currentTime value within the tolerance of +/- (2/framerate + 20ms).
3) Every video frame S[k,s] shall be rendered and the video frames shall be rendered in increasing presentation time order.
4) Video start-up delay: The start-up delay should be sufficiently low. As user agents may pre-load the first frame, the time to first frame is not relevant, but what is relevant is that once hitting play, the second frame is rendered within the considered start-up delay. In addition, there may be missing frames at start up. Hence, TR [k, x] – Ti < TSMax where x is the first detected frame change after the play() event.

8.2.5.3 Audio

If the track is an audio track, then the following additional observations are expected: the following additional observations are expected:

1) When examined as a continuous sequence of timestamped audio samples of the audio stream, the 20ms test audio samples shall be a complete rendering of the source audio track and are rendered in increasing presentation time order.

2) Audio start-up-delay: The start-up delay shall be sufficiently low: `TR[k, 1] – Ti < TSMax`.

NOTE: In this case, an "audio sample" is a sequence of 20ms of discrete sampled-data values taken at 48 kHz sample rate – i.e., 960 audio clock samples. (Compare to ISOBMFF, where an audio sample is "a compressed section of audio in decoding order" , i.e. conceptually similar to an access unit in other MPEG specifications. An ISOBMFF sample will be encoded from many uncompressed samples.)

8.2.5.4 Subtitle

None.

8.3 Random Access to Fragment

8.3.1 Background

A track is randomly accessed, and playback is started from a specific time onwards.

The random access occurs at a CMAF Fragment boundary.

8.3.2 Pre-condition

A CMAF Track is available for playback following the properties in clause 5.3.2.4 with the earliest presentation time `tf[k,i=1] = 0`. All relevant information for initialization is provided, for example through a test content manifest as defined in 5.3.4.2.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5 including the relevant output. This includes:

1) Appending the CMAF Header for the track.

2) Establishing a proper output environment.

8.3.3 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback.
2) `random_access_fragment`: Defines the fragment at which the track is randomly accessed.
3) `TSMax`: The maximum permitted startup delay set to 120ms.

   NOTE: This constraint is defined as a first approach but may be refined after running some initial tests.

8.3.4 Stimulus

For a track buffer that supports a media profile, Random Access of a CMAF Track `k` at fragment `random_access_fragment`, consisting of a sequence of CMAF Header and CMAF Fragments refers to the following actions:

- Append the CMAF Header `CH[k]` to the Source Buffer.
- Set time offset to `tf[k,i=random_access_fragment]`.
- Append CMAF Fragments `CF[k,i]` in order starting from `i=random_access_fragment`.
- Load as many CMAF fragments `CF[k,i]` starting from fragment `random_access_fragment` such that the buffer duration is at least `min_buffer_duration`.
- Once reached, initiate play-back on the media source and observe:
  - The measured time when playback is initiated is `Ti`.
  - The measured time when the first sample is rendered at time `TR[k,s1]`.
  - The measured time when sample s is rendered is time `TR[k,s]`.
- While it is not the last fragment:
  - As soon as the `buffer` is `min_buffer_duration` or below, append next fragment `CF[k,i]`.
- Stop the playback at the end of the last CMAF Fragment buffer.

8.3.5 Required Observations

8.3.5.1 General

If the above algorithm is carried out, the following observations are expected:

1) The playback duration shall match the duration of the CMAF Track, i.e., `TR[k,S]` = TR [k, s1] + `td[k] - tf[k,i= random_access_fragment]` taking into account:

- Missing starting and ending frames.
- Potential start frames being rendered before playback.
- Frozen last frame after the playback ended.

2) No sample earlier than `random_access_fragment` shall be rendered.
3) The end of the video playback shall be announced.

8.3.5.2 Video

If the track is a video track, then the following additional observations are expected: See section 8.2.5.2.

8.3.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

8.3.5.4 Subtitle

None.

8.4 Random Access to Time

8.4.1 Background

A track is randomly accessed and played back, starting from a specific time onwards.

The time for playback generally does coincide with the start of a Fragment. For testing, a time other than the start of a CMAF Fragment shall be used. Testing when accessing at a start of a CMAF Fragment is covered in clause 8.3.

8.4.2 Pre-condition

A CMAF Track is available for playback following the properties in clause 5.3.2.4 with the earliest presentation time `tf[k,i=1] = 0`. All relevant information for initialization is provided, for example through a test content manifest as defined in 5.3.4.2.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5 including the relevant output. This includes:

1) Appending the CMAF Header for the track.
2) Establishing a proper output environment.

8.4.3 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback.

2) `random_access_time`: Defines the presentation time at which the track is randomly accessed.

3) `TSMax`: The maximum permitted startup delay set to 120ms.

NOTE: This constraint is defined as the first approach but may be refined after running some initial tests.

8.4.4 Stimulus

For a track buffer that supports a media profile, Random Access of a CMAF Track `k` at the presentation time `random_access_time`, consisting of a sequence of CMAF Header and CMAF Fragments refers to the following actions:

- Append the CMAF Header `CH[k]` to the Source Buffer.
- Find the Fragment number containing the presentation time `random_access_time`: `tk[k,r]`≤ `random_access_time` and `tk[k,r+1]> random_access_time`.
- *Set time offset* to `tk[k,r]` and set the `currentTime` on the Media Element to the `random_access_time - tk[k,r]`.
- Append CMAF Fragments `CF[k,i]` in order starting from `i=r`.
- Load as many CMAF fragments `CF[k,i]` starting from fragment `is` such that the buffer duration is at least `min_buffer_duration`.
- Once reached, initiate play-back on the media source and observe:
    - The measured time when playback is initiated is `Ti`.
    - The measured time when the first sample is rendered at time `TR[k,s1]`.
    - The measured time when sample `s` is rendered is time `TR[k,s]`.
- While it is not the last fragment:
    - As soon as the `buffer` is `min_buffer_duration` or below, append next fragment `CF[k,i]`.
- Stop the playback at the end of the last Fragment in buffer.

8.4.5 Required Observations

8.4.5.1 General

If the above algorithm is carried out, the following observations are expected:

1) No sample with a presentation time less than `random_access_time` shall be rendered.
2) The playback duration shall match the duration of the CMAF Track – i.e., `TR[k,S] = TR [k, s1]` + `td[k] - random_access_time`, taking into account:
    - Missing starting and ending frames.
    - Potential start frames being rendered before playback.
    - Frozen last frame after the playback ended.

8.4.5.2 Video

If the track is a video track, then the following additional observations are expected: See clause 8.2.5.2.

8.4.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

8.4.5.4 Subtitle

None.

8.5 Switching Set Playback

8.5.1 Background

Playback of a switching set assumes that the application can switch across the fragments of different tracks without observing any temporal or spatial misalignment during playback.

8.5.2 Pre-Conditions

A CMAF Switching Set is available for playback following the properties in clause 5.3.2.4 with in total `K` tracks in the Switching Set.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5 including the relevant output. This includes:

1) Appending the CMAF Principal Header for the Switching Set.
2) Establishing a proper output environment.

8.5.3 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback.
2) `playout[i]`: Provides the CMAF track number for every fragment position `i=1,…,N`. The value shall be between 1 and K.
    - proposed playout `1, 2, …, K, K-1, …, 1, K, 1, K`, then repeat.
3) `TSMax`: The maximum permitted startup delay is set to 120ms.

    NOTE: This constraint is defined as a first approach but may be refined after running some initial tests.

8.5.4 Stimulus

For a track buffer that supports a media profile, playback of a Switching Set of a CMAF Switching Set, consisting of `K` tracks, the following applies:

- Set the `LastHeader` to the CMAF Principal Header `CH*` used of initialization.
- Set presentation time offset to `tf[k=playout[i=1],i=1]`.
- Load as many CMAF fragments `CF[k,i]` starting from fragment 1 such that the buffer duration is at least `min_buffer_duration` using the Append-Algorithm-1, outlined below:
- Once reached the `min_buffer_duration`, initiate play-back on the media source and observe:
    - The measured time when playback is initiated is `Ti`.
    - The measured time when the first sample is rendered at time `TR[s=1]`.
    - The measured time when sample `s` is rendered is time `TR[s]`.
- While it is not the last fragment:
    - As soon as the `buffer` is equal to `min_buffer_duration` or below, append next fragment `CF[k,i]` using Append-Algorithm-1, outlined below.
- Stop at the end of the last CMAF Fragment in the buffer.

Append-Algorithm-1:

- For each CMAF Fragment position `i=1,…,N`:
    - If CMAF Header `CH[k = playout[i]]!= LastHeader`:
        - Append the CMAF Header `CH[k=playout[i]]` to the Source Buffer.
        - Set `LastHeader` to `CH[k=playout[i]]`.
    - Append CMAF Fragment `CF[k=playout[i],i]`.

8.5.5 Required Observations

8.5.5.1 General

If the above algorithm is carried out, the following observations are expected:

1) The playback duration shall match the duration of the CMAF fragments as defined in playout taking into account:
    - Missing starting and ending frames.
    - Potential start frames being rendered before playback.
    - Frozen last frame after the playback ended.

8.5.5.2 Video

In addition, for video the following is expected to be observed: See clause 8.2.5.2.

8.5.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

8.6 Regular Playback of Chunked Content

8.6.1 Background

Sequential Chunked Playback refers to the case that a CMAF/WAVE track is played from the beginning by providing CMAF chunks to the source buffer after initialization.

8.6.2 Pre-condition

A CMAF Track is available for playback following the properties in clause 5.3.2.4 with the earliest presentation time `tf[k,i=1]=0`. The content is available in chunked format – i.e., multiple chunks are provided for each CMAF Fragment.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5 including the relevant output. This includes:

1) Appending the CMAF Header for the track.
2) Establishing a proper output environment.

8.6.3 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback. This value shall be smaller than `df[k,i]` of all Fragments.

8.6.4 Stimulus

For a track buffer that supports a media profile, Sequential Chunked Playback of a CMAF Track `k`, consisting of a sequence of CMAF Header and CMAF Chunks refers to the following actions:

- Append the CMAF Header `CH[k]` to the Source Buffer.
- Set time offset to `tf[k,i=1]`.
- Append CMAF Chunk `CC[k,i,j]` in order starting from `i=1`, and `j=1`, incrementing `j` first to the end and then incrementing `i` and resetting `j=1`, and so on.
- Load as many CMAF Chunks `CC[k,I,j]` starting from the first Chunk of the track such that the buffer duration is at least `min_buffer_duration` and not larger or equal to than `df[k,i=1]`.
- Once reached, initiate playback on the media source and observe:

    - The measured time when playback is initiated is `Ti`.
    - The measured time when the first sample is rendered at time `TR[k,s=1]`.
    - The measured time when sample s is rendered is time `TR[k,s]`.
- While it is not the last fragment:
    - As soon as the `buffer` is `min_buffer_duration` or below, append next Chunk `CC[k,i,j]`.
- Stop at the end of the last Chunk of track in the buffer.

8.6.5 Required Observations

8.6.5.1 General

If the above algorithm is carried out, the following observations are expected:

1) The playback duration shall match the duration of the CMAF Track, i.e., `TR[k,S] = TR[k,1] + td[k]` taking into account:
    - Missing starting and ending frames.
    - Potential start frames being rendered before playback.
    - Frozen last frame after the playback ended.

8.6.5.2 Video

If the track is a video track, then the following additional observations are expected: See section 8.2.5.2.

8.6.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

8.6.5.4 Subtitle

None.

8.7 Regular Playback of Chunked Content, non-aligned append

8.7.1 Background

Sequential Chunked Playback refers to the case that a CMAF/WAVE track is played from the beginning by providing CMAF chunks to the source buffer after initialization. Non-aligned refers to the situation where the data that is appended to the source buffer progressively in pieces that do not necessarily align with the boundaries of the CMAF chunks.

8.7.2 Pre-condition

A CMAF Track is available for playback following the properties in clause 5.3.2.4 with the earliest presentation time `tf[k,i=1]=0`.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5 including the relevant output. This includes:

1) Appending the CMAF Header for the track.
2) Establishing a proper output environment.

8.7.3 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback. This value shall be smaller than `df[k,i]` of all Fragments.
2) `TSMax`: The maximum permitted startup delay set to 120ms.

NOTE: This constraint is defined as a first approach but may be refined after running some initial tests.

8.7.4 Stimulus

For a track buffer that supports a media profile, Sequential Chunked Playback of a CMAF Track `k`, consisting of a sequence of CMAF Header and CMAF Chunks refers to the following actions:

- Append the CMAF Header `CH[k]` to the Source Buffer.
- Set time offset to `tf[k,i=1]`.
- For each `k,i` concatenate the `N` CMAF chunks `CC[k,i,j]` in order from `j=1`, incrementing `j` to the end `(j=N)` to form a chunked fragment `CF[k,i]`.
- For each `k,i` randomly choose `2N` positive non-zero integers `L` such that the sum of all `L[k,i,r]` equals the length in bytes of the chunked fragment `CF[k,i]`.
- Split the chunked fragment `CF[k,i]` into `2N` byte ranges `BR[k,i,r]` each with length `L[k,i,r]`.
- Append byte ranges `BR[k,i,r]` in order starting from `i=1`, and `r=1`, incrementing `r` first to the end (`2N`) and then incrementing `i` and resetting `r=1`, and so on.
- Load as many byte ranges `BR[k,i,r]` starting from the first byte range of the track such that the buffer duration is at least `min_buffer_duration` and not larger or equal to than `df[k,i=1]`.
- Once reached, initiate playback on the media source and observe:
  - The measured time when playback is initiated is `Ti`.
  - The measured time when the first sample is rendered at time `TR[k,s=1]`.
  - The measured time when sample s is rendered is time `TR[k,s]`.
- While it is not the last byte range:
  - As soon as the `buffer` is `min_buffer_duration` or below, append next byte range `BR[k,i,r]`.
- Stop at the end of the last byte range of track in the buffer.

8.7.5 Required Observations

8.7.5.1 General

If the above algorithm is carried out, the following observations are expected:

1) The playback duration shall match the duration of the CMAF Track, i.e. `TR[k,S]` = `TR[k,1]` + `td[k]` taking into account:
    - Missing starting and ending frames.
    - Potential start frames being rendered before playback.
    - Frozen last frame after the playback ended.

8.7.5.2 Video

If the track is a video track, then the following additional observations are expected: See clause 8.2.5.2.

8.7.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

8.7.5.4 Subtitle

None.

8.8 Playback over WAVE Baseline Splice Constraints

8.8.1 Background

Playback over splices enables the content provider to offer a sequence of Switching Sets with less restricted encoding requirements than a track. This feature is especially important in case of program changes and ad insertion.

8.8.2 Pre-conditions

Two CMAF Switching Sets are available for playback following the properties in clause 5.3.3.5. Consequently, all CMAF tracks in both Switching Sets conform to one media profile, one CMAF header exists for initialization of playback and this header can be appended to the source buffer without triggering a reinitialization of the decoding and rendering platform.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5 including the relevant output. This includes:

- Appending the CMAF Principal Header for the two Switching Sets.
- Establishing a proper output environment.

8.8.3 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback.
2) `playout[i]`: Provides the triple (Switching Set, CMAF track number, Fragment number) for every playout position `i=1,…,N` that is to be played out according to the attached test content configuration.

NOTE: A configuration file test.json is attached, and also accessible online at https://github.com/cta-wave/dpctf-tests/blob/master/test-config.json.

8.8.4 Stimulus

For a track buffer that supports a media profile, playback of two WAVE Baseline Splice Constraints Switching Sets of a CMAF Switching Set, consisting of `K` tracks, the following applies:

- Set the `LastHeader` to the CMAF Principal Header `CH*` used of initialization.
- Set presentation time offset to `tf[k=playout[i=1],i=1]`.
- Load as many CMAF fragments `CF[k,i]` starting from fragment 1 such that the buffer duration is at least `min_buffer_duration` using the Append-Algorithm-2, outlined below.
- Once reached the `min_buffer_duration`, initiate play-back on the media source and observe:
    - The measured time when playback is initiated is `Ti`.
    - The measured time when the first sample is rendered at time `TR[s=1]`.
    - The measured time when sample s is rendered is time `TR[s]`.
- While it is not the last fragment:
    - As soon as the `buffer` is equal to `min_buffer_duration` or below, append next fragment `CF[k,i]` using Append-Algorithm-2, outlined below.
- Stop at the end of the last CMAF Fragment in the buffer.

Append-Algorithm-2:

- For each CMAF Fragment position `i=1,…,N`:
    - If CMAF Header `CH[k = playout[i]]!= LastHeader`:
        - Append the CMAF Header `CH[k=playout[i]]` to the Source Buffer.
        - Set `LastHeader` to `CH[k=playout[i]]`.
    - Append CMAF Fragment `CF[k=playout[i],i]`.

8.8.5 Required Observations

8.8.5.1 General

If the above algorithm is carried out, the following observations are expected:

1) The playback duration shall match the duration of the CMAF fragments as defined in playout taking into account:
    - Missing starting and ending frames.
    - Potential start frames being rendered before playback.
    - Frozen last frame after the playback ended.

8.8.5.2 Video

If the track is a video track, then the following additional observations are expected: See caluse 8.2.5.2.

8.8.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

8.9 Out-Of-Order Loading

8.9.1 Background

Out of order loading refers to the case that a CMAF/WAVE track is played from the beginning by providing CMAF fragments to the source buffer after initialization, but CMAF fragments are loaded out of order within the buffer duration.

8.9.2 Pre-condition

A CMAF Track is available for playback following the properties in clause 5.3.2.4 with the earliest presentation time `tf[k,i=1]=0`.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5 including the relevant output. This includes:

1) Appending the CMAF Header for the track.
2) Establishing a proper output environment.

8.9.3 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback.
2) `max_buffer_duration`: Expresses the duration of media that can always be accommodated in the Source Buffer.

3) `loading[i]`: provides the CMAF Fragment number that is loaded at step `i`, constrained such that `(MAX(i-loading[i]) + MAX(loading[i]-i)) * MAX(df[k,i]) < max_buffer_duration.`

   NOTE: A configuration file test.json is attached, and also accessible online at https://github.com/cta-wave/dpctf-tests/blob/master/test-config.json.

The parameters need be such that in any case the loaded CMAF Fragment needs to be in the playout buffer.

8.9.4 Stimulus

For a track buffer that supports a media profile, Sequential Playback of a CMAF Track `k`, consisting of a sequence of CMAF Header and CMAF Fragments refers to the following actions:

- Append the CMAF Header `CH[k]` to the Source Buffer.
- Set presentation time offset to `tf[k,i=1]`.
- Append CMAF Fragments `CF[k,loading[i]]` in order starting from `i=1`.
- Load as many CMAF fragments `CF[k,loading[i]]` starting from index 1 such that the contiguous buffered duration is at least `min_buffer_duration`.
- Once reached, initiate play-back on the media source and observe:
  - The measured time when playback is initiated is `Ti`.
  - The measured time when the first sample is rendered at time `TR[k,s=1]`.
  - The measured time when sample s is rendered is time `TR[k,s]`.
- While it is not the last fragment:
  - As soon as the contiguous buffered duration is `min_buffer_duration` or below, append next fragment `CF[k,loading[i]]`.
- Stop at the end of the last CMAF Fragment in the buffer .

8.9.5 Required Observations

8.9.5.1 General

If the above algorithm is carried out, the following observations are expected:

1) The playback duration shall match the duration of the CMAF Track – i.e., `TR[k,S] = TR[k,1] + td[k]` taking into account:
   - Missing starting and ending frames.
   - Potential start frames being rendered before playback.
   - Frozen last frame after the playback ended.

8.9.5.2 Video

If the track is a video track, then the following additional observations are expected: See clause 8.2.5.2.

8.9.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

8.10 Overlapping Fragments

8.10.1 Background

Playback of overlapping fragments assumes that the application can overwrite the buffer with other fragments. This is for example useful in case a fragment is loaded in lower quality but may then be replaced by a better-quality fragment.

8.10.2 Pre-condition

A CMAF Switching Set is available for playback following the properties in clause 5.3.2.5 with in total `K` tracks in the Switching Set.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5 including the relevant output. This includes:

1) Appending the CMAF Header for the track.
2) Establishing a proper output environment.

8.10.3 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains during the playback. This value shall be equal or larger than `2max(df[k,i])` for all `i` and `k`.

2) `playout[i]`: Provides the CMAF track number for every fragment position `i=1,…,N`. This value shall be between `1` and `K`. The playout instructions follow the test configuration.

NOTE: A configuration file test.json is attached, and also accessible online https://github.com/cta-wave/dpctf-tests/blob/master/test-config.json.

8.10.4 Stimulus

For a track buffer that supports a media profile, Overlapped Fragment Playback of a CMAF Switching Set, consisting of `K` tracks refers to the following actions, starting with `k=1` for the track with lowest quality to `k=K` with the highest quality:

- Set the `LastHeader` to the CMAF Principal Header `CH*` used of initialization.
- Set presentation time offset to `tf[k=playout[i=1],i=1]`.

- Load as many CMAF fragments `CF[k,i]` starting from fragment 1 such that the buffer duration is at least `min_buffer_duration` using the Append-Algorithm-3, outlined below.
- Once reached the `min_buffer_duration`, initiate play-back on the media source and observe:
    - The measured time when playback is initiated is `Ti`.
    - The measured time when the first sample is rendered at time `TR[s=1]`.
    - The measured time when sample s is rendered is time `TR[s]`.
- While it is not the last fragment:
    - As soon as the `buffer` is equal to `min_buffer_duration` or below, append next fragment `CF[k,i]` using Append-Algorithm-3, outlined below.
- Stop at the end of the last CMAF Fragment in the buffer.

Append-Algorithm-3:

- For each CMAF Fragment position `i=1,…,N`
    - If CMAF Header `CH[k = playout[i]]!= LastHeader`
        - Append the CMAF Header `CH[k=playout[i]]` to the Source Buffer
        - Set `LastHeader` to `CH[k=playout[i]]`
    - Append CMAF Fragments `CF[k=playout[i],i-1]` and `CF[k=playout[i],i]`

8.10.5 Required Observations

8.10.5.1 General

If the above algorithm is carried out, the following observations are expected:

1) The playback duration shall match the duration of the CMAF Track – i.e., `TR[k,S] = TR[k,1] + td[k]` taking into account:
    - missing starting and ending frames.
    - potential start frames being rendered before playback.
    - frozen last frame after the playback ended.

8.10.5.2 Video

In addition, for video the following is expected to be observed: See clause 8.2.5.2.

8.10.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

8.11 Full Screen Playback of Switching Sets

8.11.1 Background

Fullscreen playback ensures that the device provide proper scaling and letter boxing. If the device display matches the content aspect ratio, then the entire screen is expected to be filled. If the display does not match the content aspect ratio, then it is expected that all content is shown in the largest possible size without changing the aspect ratio of the content.

8.11.2 Pre-conditions

Same as documented in clause 8.5.2.

Output is set to fullscreen.

8.11.3 Parameters and Variants

Same as documented in clause 8.5.3.

8.11.4 Stimulus

Same as documented in clause 8.5.4.

8.11.5 Required Observations

8.11.5.1 General

If the above algorithm is carried out, the following observations are expected:

1) The playback duration shall match the duration of the CMAF fragments as defined in playout taking into account:
    - Missing starting and ending frames.
    - Potential start frames being rendered before playback.
    - Frozen last frame after the playback ended.

8.11.5.2 Video

If the track is a video track, then the following additional observations are expected: See clause 8.2.5.2.

8.11.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

8.12 Playback of Encrypted Content

8.12.1 Introduction

Playback of encrypted content requires that a device under test contain a Content Decryption Module (CDM) that performs two functions:

1. Decryption of media samples encrypted with Common Encryption (CENC) scheme ‘cenc’ or ‘cbcs’, followed by normal decoding and rendering equivalent to unencrypted content.

2. Exchange of license requests and licenses containing keys between the CDM and a license server, e.g. by the test runner through the W3C Encrypted Media Extension API (EME). See 7.2.2.

Requesting and downloading a license is a function of a player or test runner and a license server, not the device under test. The test runner and license server are not being tested but must function correctly to securely acquire keys that are cryptographically bound to the CDM for the purpose of testing device decryption.

The security and functionality of the DRM system is considered out of the scope of WAVE tests, other than to enable media decoding and rendering of encrypted content test cases. It is assumed that each CDM is fully specified and implemented according to the requirements of the DRM provider.

8.12.2 Encrypted Content Use Cases

8.12.2.1 Introduction

The following use cases combine four methods of key management with the other media playback use cases and observations specified elsewhere in this document:

1. A single Track or Presentation with one key.
2. A single Presentation with different audio and video keys.
3. Multiple Presentations sequenced in a Program, encrypted and clear. One license, but interspersed clear content – e.g., an encrypted show interspersed with unencrypted ads.
4. Multiple Presentations sequenced in a Program that require different licenses (“license rotation”). Licenses must be downloaded and updated without interrupting playback. The Playlist should include license tags prior to the first dependent segment tag to simulate just-in-time license processing.

Playback Observation: Playback observations shall equal the equivalent unencrypted test case, except where DRM security requirements prohibit normal playback.

For instance, playback of UHD or HD content could be blocked or subsampled to lower resolution on analog video interfaces or digital interfaces with too low an HDCP version number. That is correct playback behavior under DRM control, given a device in a particular system configuration with those license restrictions.

Tests, such as trick play, are valid for testing parser and CDM handling of encryption and key changes out of sequence. However, they should test media pipeline functions, not the speed of license acquisition from the test license server. The test runner will not respond to unidentified keys found in content (by handling fired "need key" events and initiating license downloads), but instead test playlists shall include tags instructing the test runner to download the necessary keys in advance of their being needed for decryption.

WAVE tests are not defined at this time for "key rotation" and hierarchical licenses, which can be stored in ‘pssh’ boxes in Fragments. Key rotation allows keys to change over time without requiring a CDM-unique out of band license downloaded for each key change. Hierarchical licenses also enable authorizing a channel, subscription, etc. with a single parent license that is unique to each CDM but authorizes child licenses that are different for each piece of content but readable by all CDMs, so can be broadcast in the content. These are currently considered functions and tests covered by each DRM system.

8.12.2.2 Signaling Requirements and Capability Selection

A test playlist shall identify the encryption format applied and one or more DRM systems, DRM initialization data, and license server locations able to provide the necessary decryption keys.

The playlist encryption information shall match the `default_KID` and Common Encryption scheme found in the Track Encryption Box (‘`tenc`’) in the CMAF Headers that follow it. If different keys are used in different Switching Sets – e.g., different keys for audio and video, or SD, HD, and UHD video – then the playlist shall identify the `default_KID` needed prior to the first instance of appending a segment from that Switching Sets. Playlist tags are sequentially processed, so the location of a license tag determines the relative timing of the license request relative to a Header or segment request.

The test runner uses playlist tags to identify the encryption scheme used and DRM licenses offered so the test runner can determine if a device can decrypt the signaled decryption scheme (‘cenc’ or ‘cbcs’) and supports one of the listed DRM systems (e.g., Widevine, FairPlay, or PlayReady). The test runner is expected to automatically query device DRM system capability, and request licenses in a device supported DRM system. See clause 7.

The test runner is only expected to respond to tags in the playlist, not Header information such as the sample entry or ‘`tenc`’ box, or Fragment information such as sample groups and their descriptions. Headers, sample groups, and sample auxiliary information are used by the device’s media pipeline to determine what scheme to decrypt with what keys and byte ranges when fully parsing and processing the media. But the test runner only needs to parse the playlist to fetch the correct license before appending dependent media segments.

The test runner and license server implement an authorization protocol for WAVE testing using an access token or other means to directly authenticate the test runner and authorize license downloads. The license server(s) maintain a secure key database indexed by KID, so license requests need not convey the keys in the license requests (typically done in real world applications using JSON web tokens attached by Auth servers forwarding the license request to the license server).

8.12.2.3 Playback Requirements

Devices shall decrypt at least one CENC scheme (‘cenc’ or ‘cbcs’) using at least one DRM system.

Devices should support decryption and DRM capability discovery through the EME API or similar native API.

Devices should indicate which encryption schemes and DRM systems are supported for each Media Profile in the case where only some combinations of Media Profile, decryption scheme, and DRM system are supported in combination. See 7.2.2.

8.12.3 Test Pre-condition

A CMAF Track is available for playback following the properties in clause 5.3.2.4. The track is encrypted with Common Encryption ‘`cenc`’ or ‘`cbcs`’ encryption scheme and a key identified by the key identifier `default_KID`. There is a license available for the track and a license server. The device needs to include a CDM and APIs that a test runner can use to negotiate a license with the license server as defined in clause 7.2.2.

A capability check for the specific encryption scheme is done and was successful, as defined in clause 7.2.2.

A license suitable for the supported device capabilities is selected. A license is acquired and available before the Media Source is established as defined in clause 7.2.2.

A Media Source is established as defined in clause 6.2.

The media capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5.

8.12.4 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback <needs >.
2) Encryption scheme (‘cenc’|’cbcs’) and default_KID.
3) One or more DRM SystemID and InitData tags and a listed or known license server URL.

8.12.5 Stimulus

After the necessary licenses are acquired and the secure media pipeline initialized:

For a track buffer that supports a media profile, Sequential Playback of a CMAF Track `k`, consisting of a sequence of CMAF Header and CMAF Fragments refers to the following actions:

- Append the CMAF Header `CH[k]` to the Source Buffer.
- Set presentation time offset to 0.
- Append CMAF Fragments CH[i] in order starting from i=1 .
- Load as many fragments starting from fragment 1 such that the buffer is at least `min_buffer_duration`.
- Start-play time T1.
    - The first sample is rendered at time T2.
- While it is not the last fragment, do the loop: As soon as the `buffer (need to measure)` is `min_buffer_duration` or below, add next fragment.
- Stop at the end of the buffer.

8.12.6 Required Observations

8.12.6.1 General

If the above algorithm is carried out, the following observations are expected:

1) The playback duration shall match the duration of the CMAF Track – i.e., `TR[k,S] = TR[k,1] + td[k]` taking into account:
    - Missing starting and ending frames.
    - Potential start frames being rendered before playback.
    - Frozen last frame after the playback ended.

8.12.6.2 Video

If the track is a video track, then the following additional observations are expected: See section 8.2.5.2.

8.12.6.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

8.13 Restricted Splicing of Encrypted Content

8.13.1 Background

Unencrypted samples and key changes are signaled by sample groups and sample group descriptions in each Fragment that contain keys or encryption state different from the default signaled in the `'tenc'` box of the CMAF Principal Header.

A CMAF Principal Header for each Switching Set initializes a media pipeline sufficient to decrypt, decode, and display all Fragments that follow.

Multiple Track Switching Sets are available conforming to CMAF single initialization constraints.

The test playlist requests licenses with the necessary keys in advance.

Time discontinuities are signaled by a discontinuity tag, and the presentation time offset set to the 'tfdt' BaseMediaDecodeTime.

8.13.2 Pre-conditions

Content according to the above conditions is available.

8.13.3 Required Observations

A test playlist shall first initialize a secure media pipeline for each media component – i.e., one that includes a CDM and protected path in response to initializing decryption as defined in clause 7.2.2.

One set of licenses based on the default_KID in each Principal Header shall be downloaded and shall be sufficient to decrypt the duration of the Program without additional license downloads.

Common Encryption `'seig'` sample groups shall be processed without interruption of the secure media pipeline.

8.13.4 Playback Observation:

The same as 8.8.5.

NOTE: Initialization and license downloading rely on the playlist and test runner. Signaling of the encryption scheme, encryption options, subsample byte ranges, default KID, sample group KID, initialization vectors, etc. are all signaled by Common Encryption information in ISOBMFF boxes that enable decryption by a CDM and secure media pipeline without reliance on the playlist or test runner (other than appending Headers at splices where encryption changes).

8.14 Sequential Playback of Encrypted and Non-Encrypted Baseline Content

8.14.1 Background

A transition from encrypted to non-encrypted content (and vice versa) is an interesting test. For instance, this can be important in the context of ad-insertion. Two main cases exist:

Case 1

- Append init segment of encrypted content.
- Append media segment of encrypted content.

- Call `changeType()`.
- Append init segment of unencrypted content.
- Append media segment of unencrypted content.
- Playback works.

Case 2

- Append init segment of unencrypted content.
- Append media segment of unencrypted content.
- Call `changeType()`.
- Append init segment of encrypted content.
- Append media segment of encrypted content.
- MSE error.

8.14.2 Conditions

Test vectors shall include a sequence of Presentations conforming to WAVE Baseline splice constraints, including both encrypted and unencrypted Presentations using either `'cenc'` or `'cbcs'` scheme, but not both.

A test playlist shall append a Header and signal presentation time offset discontinuity at each splice point between Presentations.

8.14.3 Stimulus

A test playlist shall first initialize a secure media pipeline for each media component – i.e., one that includes a CDM and protected path in response to initializing decryption as defined in clause 7.2.2.

Unencrypted Presentations shall be processed without reconfiguration of the secure media pipeline, but splices shall include Header appends in the test playlist, i.e., changing between encrypted and clear content as signaled by a 'tenc' box or the lack thereof, or change in the default_KID, which requires a license tag to tell the runner to fetch a different license.

> NOTE: Initialization and license downloading rely on the playlist and test runner. Signaling of the encryption scheme, encryption options, subsample byte ranges, default KID, sample group KID, initialization vectors, etc. are all signaled by Common Encryption information in ISOBMFF boxes that enable decryption by a CDM and secure media pipeline without reliance on the playlist or test runner (other than appending Headers at splices where encryption changes).

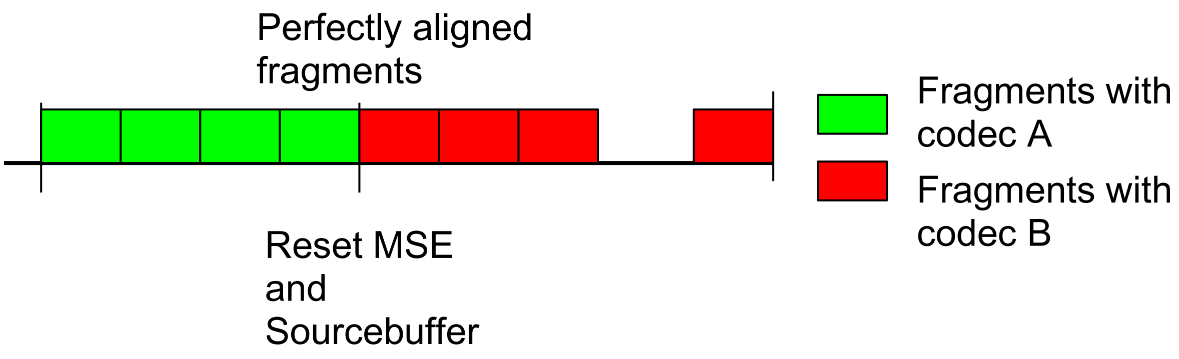
8.14.4 Required Observation

The same as 8.8.5.

8.15 Source Buffer Re-Initialization (without changeType)

8.15.1 Background

This test provides a stimulus in case a source buffer needs to be re-established, for example because of a codec change or a codec parameter change. This frequently happens when the player is transitioning between main content and advertisements or when the user manually selects a new track. This test resets the MSE in order to transition between the different CMAF Switching Sets.



8.15.2 Pre-Condition

Two CMAF Switching Sets are available for playback. They do not belong to the same media profile.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5 including the relevant output. This includes:

1. Establishing a proper output environment.

The CMAF fragments of the required CMAF Switching Sets and CMAF Switching Tracks are aligned and continuous (compare to fragment 4 and fragment 5 in the example above).

8.15.3 Parameters and Variants

The playback has the following parameters:

- `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback.
- `playout[i]`: Provides an array of entries for every playout position i=1,...,N that is to be played out. Each entry is a triple structured {s=Switching Set, k=CMAF track number, f=Fragment number}.
- `mse_reset_tolerance`: Expresses the tolerance that is acceptable for a single MSE reset. This value is defined in milliseconds.

8.15.4 Parameter and Variants restrictions

- The accumulated duration of the fragments of the first CMAF switching set defined in *playout* shall be larger than *`min_buffer_duration`*. This avoids scenarios in which an MSE reset is triggered before the playback is initiated.

8.15.5 Stimulus

- Set `SourceBuffer.timestampOffset` to `–tf[playout[1].k,playout[1].f]` *(Earliest presentation time of first fragment to be played)*.
- Set `mseResetCounter` to 0.
- Load as many CMAF fragments starting from fragment 1 such that the buffer duration is at least `min_buffer_duration` using the Append-Algorithm-2, outlined below.
- Once the `min_buffer_duration` is reached, initiate playback on the media source and observe:
  - While it is not the last fragment:
    - As soon as the buffer is equal to `min_buffer_duration` or below, append the next fragment `CF[k,i]` using Append-Algorithm-2, outlined below.
  - Stop at the end of the last CMAF Fragment in the buffer.

Append-Algorithm-2:

For each CMAF Fragment position `i=1,...,N`:

- If CMAF Header `CH[playout[i].k]!= LastHeader`
  - Append the CMAF Header `CH[playout[i].k]` to the Source Buffer. Set `LastHeader` to `CH[playout[i].k]`.
  - If the two fragments i and i+1 are not codec compatible:
    - Wait for all fragments to be played and the current play position is equal to `tf[playout[i].k,i] - tf[playout[1].k,playout[1].f]` (*empty buffer since current play position reached the start of Fragment i*)
    - Remove the `SourceBuffer` from the `MediaSource`.
    - Detach the `MediaSource`.
    - Attach a new `MediaSource`.
    - Add a new `SourceBuffer` to the `MediaSource`.
    - Set `SourceBuffer.timestampOffset` to – `tf[playout[1].k,playout[1].f]` (Earliest presentation time of first fragment to be played).
    - Seek to `tf[playout[i].k,i] - tf[k=playout[1].k,playout[1].f]` (*start of Fragment i*).
    - Increase `mseResetCounter` by 1.
- Append CMAF Fragment `CF[k=playout[i],i]`.

8.15.6 Required Observations

8.15.6.1 General

If the Append-Algorithm-2 is carried out, the following observations are expected:

1) The playback duration shall match the duration of the CMAF fragments as defined in playout taking into account:
    - `mse_reset_tolerance`.
    - Missing starting and ending frames.
    - Potential start frames being rendered before playback.
    - Frozen last frame after the playback ended.

8.15.6.2 Video

If the track is a video track, then the following additional observations are expected: See section 8.2.5.2.

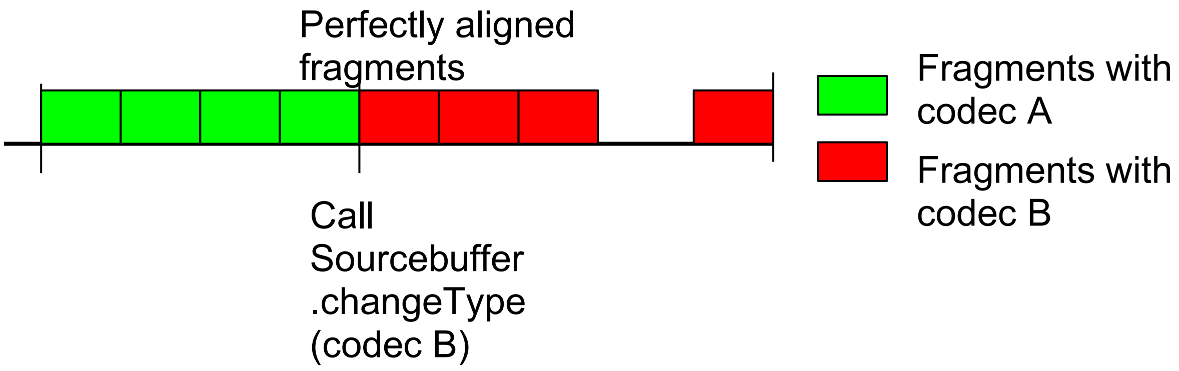
8.15.6.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

8.16 Source Buffer Re-Initialization (with changeType)

8.16.1 Background

This test provides a stimulus in case a source buffer needs to be re-established for example because of a codec change or a codec parameter change. This frequently happens when the player is transitioning between main content and advertisements, or when the user manually selects a new track. This test uses the changeType method on the `SourceBuffer` object before a codec change. (https://w3c.github.io/media-source/#dom-SourceBuffer-changetype).


Perfectly aligned fragments
Fragments with codec A
Fragments with codec B
Call Sourcebuffer .changeType (codec B)

8.16.2 Pre-Condition

Two CMAF Switching Sets are available for playback. They do not belong to the same media profile.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5 including the relevant output. This includes:

1. Establishing a proper output environment.

The CMAF fragments of the required CMAF Switching Sets and CMAF Switching Tracks are aligned and continuous (compare to fragment 4 and fragment 5 in the example above)

8.16.3 Parameters and Variants

The playback has the following parameters:

- `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback.
- `playout[i]`: Provides the triple {Switching Set, CMAF track number, Fragment number} for every playout position `i=1,...,N` that is to be played out.

8.16.4 Parameter and Variants restrictions

- The accumulated duration of the fragments of the first CMAF switching set defined in *playout* shall be larger than `min_buffer_duration`. This avoids scenarios in which `changeType()` is triggered before the playback is initiated.

8.16.5 Stimulus

- Set `SourceBuffer.timestampOffset to -tf[playout[1].k,playout[1].f]` *(Earliest presentation time of first fragment to be played)*.
- Load as many CMAF fragments `CF[k,i]` starting from fragment 1 such that the buffer duration is at least `min_buffer_duration` using the Append-Algorithm-3, outlined below.
- Once the `min_buffer_duration` is reached, initiate playback on the media source and observe:
  - While it is not the last fragment:
    - As soon as the buffer is equal to `min_buffer_duration` or below, append the next fragment `CF[k,i]` using Append-Algorithm-3, outlined below.
  - Stop at the end of the last CMAF Fragment in the buffer.

Append-Algorithm-3:

For each CMAF Fragment position i=1,...,N:

- If CMAF Header `CH[playout[i].k]!= LastHeader`:
  - If the two fragments i and i+1 are not codec compatible,
    - Call `changeType(mimeType)` on the `SourceBuffer`.
  - Append the CMAF Header `CH[playout[i].k]` to the Source Buffer. Set LastHeader to `CH[playout[i].k]`.
- Append CMAF Fragment `CF[k=playout[i],i]`.

8.16.6 Required Observations

8.16.6.1 General

If the Append-Algorithm-3 is carried out, the following observations are expected:

1) The playback duration shall match the duration of the CMAF fragments as defined in playout taking into account:
    - Missing starting and ending frames
    - Potential start frames being rendered before playback
    - Frozen last frame after the playback ended

8.16.6.2 Video

If the track is a video track, then the following additional observations are expected: See section 8.2.5.2.

8.16.6.3 Audio

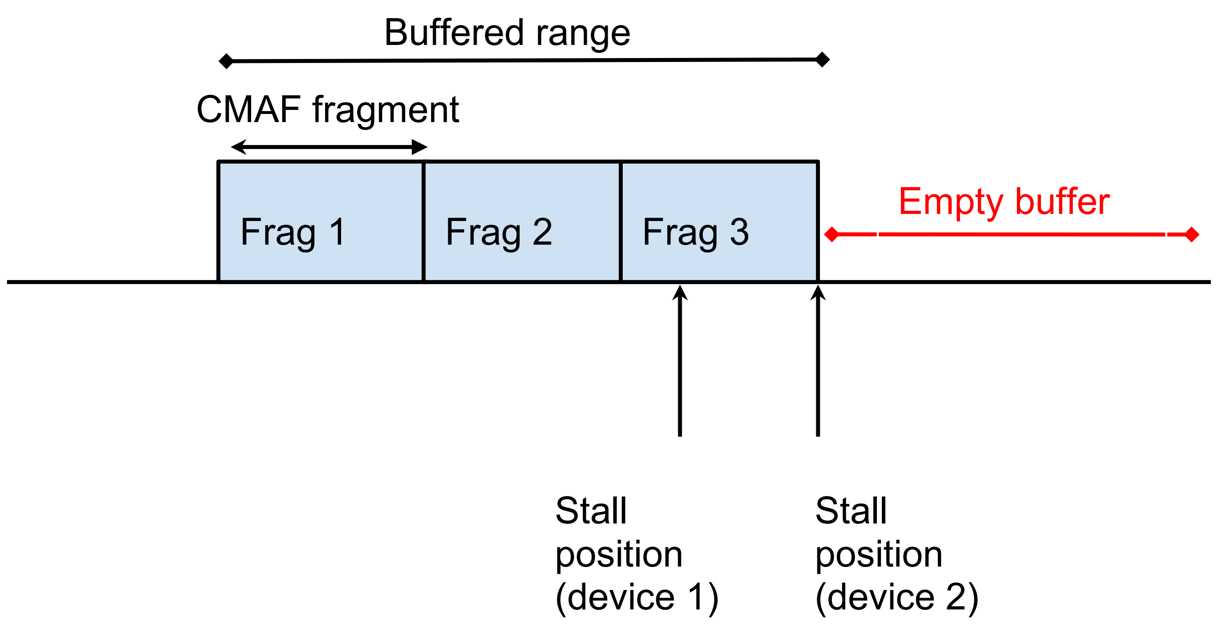
If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

8.17 Buffer Underrun and Recovery

8.17.1 Background

This case addresses situations in which the buffer in play mode runs out of media and is recovering from this situation. Reasons for that include a bandwidth drop on the client side, delayed signaling of CMAF Fragments in the manifest or delayed availability of CMAF Fragments and CMAF chunks.

> NOTE: Depending on the platform and the target device, the buffer level at which the stall event is dispatched might vary. As an example, on one device it might be required to have at least 0.5 seconds of content in the buffer before the video element transitions to the playing state again. Other devices only require a single frame to transition to the playing state.



8.17.2 Pre-Condition

A CMAF Track is available for playback following the properties in clause 5.3.2.4.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5, including the relevant output. This includes:

1. Appending the CMAF Header for the track.
2. Establishing a proper output environment.

8.17.3 Parameters and Variants

- `min_buffer_duration`: Expresses the initial duration of the `SourceBuffer`. Once this duration is reached the playback shall be triggered (call to video.play()).
- `waiting_duration`: Time to wait after a "stall" event dispatched by the browser. After the `waiting_duration`, new data is appended to the buffer in order to transition out of the stalling state.
- The waiting state is signaled by the application to the observation framework.

8.17.4 Stimulus

- Append the CMAF Header `CH[k]` to the Source Buffer.
- Set `SourceBuffer.timeOffset` to -`tf[k,1]`. This is only required if `tf[k,1]`!=0.

- Load as many CMAF fragments `CF[k,i]` starting from fragment 1 such that the buffer duration is equal or larger than `min_buffer_duration`: Let `CF[k,t]` be the fragment for which *tf[k,t] + df[k,t]* - *`tf[k,1]`* >= `min_buffer_duration`.
- Initiate play-back on the media source and observe:
  - Playback should start.
  - Playback should stall shortly before or exactly when reaching the end of the buffer duration.
- Wait until the video element transitions into the "waiting" state.
- Set timeout duration to `waiting_duration`
- After the timeout has elapsed continue appending from fragment t and observe:
  - Playback should resume. The video element shall transition to "playing" state.
- Playback should end when all samples of the last fragment are rendered.

8.17.5 Required Observations

8.17.5.1 General

1) The playback duration shall match the duration of the CMAF Track, i.e. `TR[k,S]` = `TR[k,1]` + `td[k]` taking into account:
   - Missing starting and ending frames.
   - Potential start frames being rendered before playback.
   - Frozen last frame after the playback ended.
   - `waiting_duration` as the time to wait before appending new data to the buffer after a stall event.

8.17.5.2 Video

If the track is a video track, then the following additional observations are expected: See section 8.2.5.2.

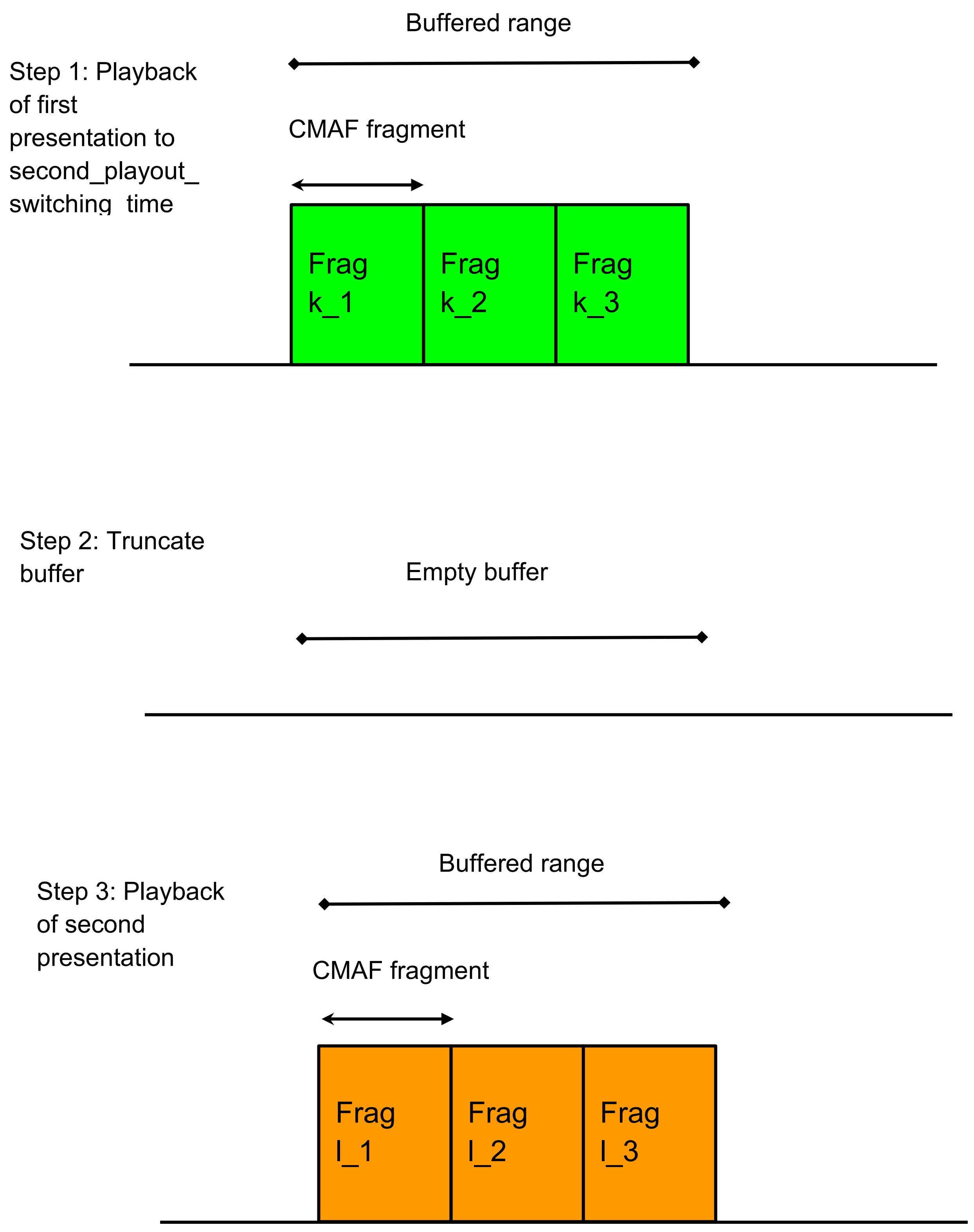
8.17.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

8.18 Truncated Playback and Restart

8.18.1 Background

This test addresses the complete truncation of the existing buffer in order to start playback of a new CMAF presentation:


Buffered range
Step 1: Playback of first presentation to second_playout_switching time
CMAF fragment
Frag k_1
Frag k_2
Frag k_3
Step 2: Truncate buffer
Empty buffer
Step 3: Playback of second presentation
Buffered range
CMAF fragment
Frag l_1
Frag l_2
Frag l_3

8.18.2 Pre-Condition

Two CMAF Switching Sets are available for playback. Both Switching Sets shall use the same codec settings in order to avoid resets of the `MediaSource` and `SourceBuffer`.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5 including the relevant output. This includes:

1. Appending the CMAF Header for the track.
2. Establishing a proper output environment.

The CMAF fragments of the required CMAF Switching Sets and CMAF Switching Tracks are aligned and continuous.

8.18.3 Parameters and Variants

- `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback.
- `playout[i]`: Provides the triple (Switching Set, CMAF track number, Fragment number) for every fragment in the first presentation.
- `second_playout_switching_time`: Time to switch from the first to the second presentation. This shall be less than the duration of the first presentation. (This prevents the device entering a waiting state).
- `second_playout[i]`: Provides the triple (Switching Set, CMAF track number, Fragment number) for every fragment in the second presentation.
- `TSMax`: The maximum permitted startup delay of the second presentation, set to 120ms.

  NOTE: TSMAx in this case refers to the startup of the *second* presentation in contrast to the other cases where it typically refers to the initial startup. The value is set to identical values of initial startup.

8.18.4 Stimulus

- Append the CMAF Header `CH[k]` to the Source Buffer.
- Set `SourceBuffer.timeOffset` to `-tf[k,1]`.
- Load as many CMAF fragments `CF[k,i]` as defined in "playout" starting from fragment 1 such that the buffer duration is equal or larger than `min_buffer_duration`.
- Initiate play-back on the media source and observe:
  - Playback should start.
- While it is not the last fragment:
  - As soon as the buffer is `min_buffer_duration` or below, append next fragment `CF[k,i]` as defined in "playout".
- Wait for play position to reach "second_playout_switching_time".
- Truncate the buffer from `br[0].start` to `br[0]`.end (since the segments are aligned there will only be one range object for the `SourceBuffer`).
- Set the `currentTime` of the video element to 0.

- Append the CMAF Header CH[l] to the Source Buffer.
- Set `SourceBuffer.timeOffset` to `-tf[l,1]`.
- Load as many CMAF fragments `CF[l,i]` as defined in "second_playout" starting from fragment 1 such that the buffer duration is equal or larger than `min_buffer_duration`.
- Initiate play-back on the media source and observe:
  - Playback should start.
- Continue appending CMAF [l] fragments as defined in "second_playout" to the buffer.
  - Playback continues to the end of the media.

8.18.5 Required Observations

8.18.5.1 General

1) The first presentation playback duration is equal to or greater than the duration of the "second_playout_switching_time" and the second presentation playback duration shall match the second presentation duration, taking into account:
   - Missing starting and ending frames.
   - Potential start frames being rendered before playback.
   - Potential extra first presentation frames being rendered after the `"second_playout_switching_time"`.
   - Frozen last frames after the `"second_playout_switching_time"` and the playback ended.

8.18.5.2 Video

If the track is a video track, then the following additional observations are expected:

1) Every video frame S[k,s] shall be rendered such that it fills the entire video output window following the properties in clause 5.2.2.
2) For each of the individual presentations, the presented sample shall match the one reported by the `currentTime` value within the tolerance of `+/- (2/framerate + 20ms)`.
3) Every video frame S[k,f] up to the "second_playout_switching_time" and every video frame S[l,f] shall be rendered and the video frames shall be rendered in increasing presentation time order. Video frames from the first presentation are rendered only once. Taking into account:
   - Potential extra first presentation frames being rendered after the "second_playout_switching_time".
   - Frozen last first presentation frame or blank frames after the "second_playout_switching_time".
4) The start-up delay should be sufficiently low. The delay between `"representation_change"` and presenting the first frame of the second presentation should be sufficiently low. See section 8.2.5.2.

8.18.5.3 Audio

If the track is an audio track, then the following additional observations are expected:

1) When examined as a continuous sequence of timestamped audio samples of the audio stream up to the "second_playout_switching_time", the `20 ms` test samples shall be a complete rendering of the source audio track and are rendered in increasing presentation time order.
2) Audio samples from the first presentation are rendered only once. Taking into account:

- Potential extra first presentation audio samples being rendered after the "second_playout_switching_time".
- Frozen last first presentation audio sample or blank samples after the "second_playout_switching_time".

8.19 Low-Latency (1): Initialization

8.19.1 Background

After initialization of the media decoder, it is important that as soon as the first media is added, decoding and presentation is started as quickly as possible. Initialization may be done ahead of time.

One of the obvious tests is to compare content, with an artificial delay between when the initialization segment is appended and when the media data is appended. Then measure the time from when the first media data is appended to when the first content is visible.

8.19.2 Pre-Condition

A CMAF Track is available for playback following the properties in clause 5.3.2.4 with the earliest presentation time `tf[k,1]`=0.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5 including the relevant output. This includes:

1) Appending the CMAF Header for the track.
2) Establishing a proper output environment.

8.19.3 Parameters and Variants

- `waiting_duration`: Expresses the waiting duration in milliseconds between appending the CMAF header and appending the first CMAF chunk. The waiting state is signaled via a QR code by the application for usage by the observation framework.
- render_threshold: Expresses the maximum allowed time between successfully appending the first CMAF fragment and the first media sample being visible or audible.
- `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains during the startup phase and during the playback.

8.19.4 Stimulus

- Append the CMAF Header *`CH[k]`* to the Source Buffer.
- Set `SourceBuffer.timestampOffset` to -*`tf[k,1]`*.
- Initiate play-back on the media source and observe:
  - Video element should throw a waiting event to indicate that the video is waiting for more data
  - The video element's *readystate* property transitions to the HAVE_METADATA state (Enough of the media resource has been retrieved that the metadata attributes are initialized).

- Wait for the duration of `waiting_duration`.
- Load as many CMAF Chunks CC[k,i,j], starting from the first chunk of the track such that the buffer duration is at least `min_buffer_duration`. Once `min_buffer_duration` has been reached or exceeded stop the appending operation and show a notification for observation (QR code).
- Observe: Playback should start.
- Append the rest of the CMAF chunks of CMAF track k.

8.19.5 Required Observations

8.19.5.1 General

1) The playback duration shall match the duration of the CMAF Track, i.e., `TR[k,S] = TR[k,1] + td[k]` taking into account:
   - Missing starting and ending frames.
   - Potential start frames being rendered before playback.
   - Frozen last frame after the playback ended.
2) Measure the time between the successful appending of the first CMAF chunk that exceeded `min_buffer_duration` and the first media sample being visible or audible. This value shall be compared against render_threshold.

8.19.5.2 Video

If the track is a video track, then the following additional observations are expected: See section 8.2.5.2.

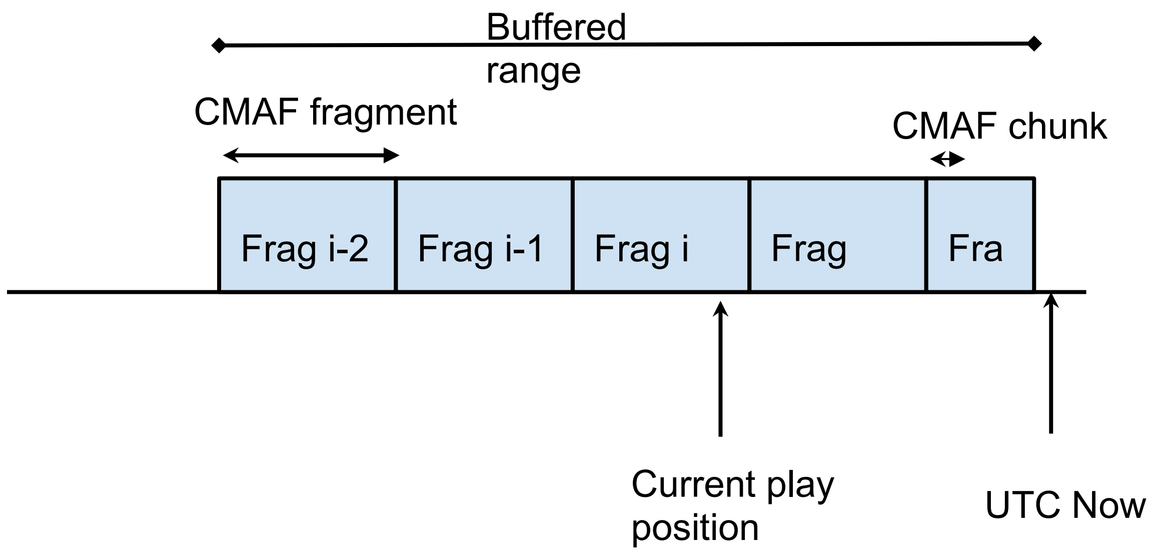
8.19.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

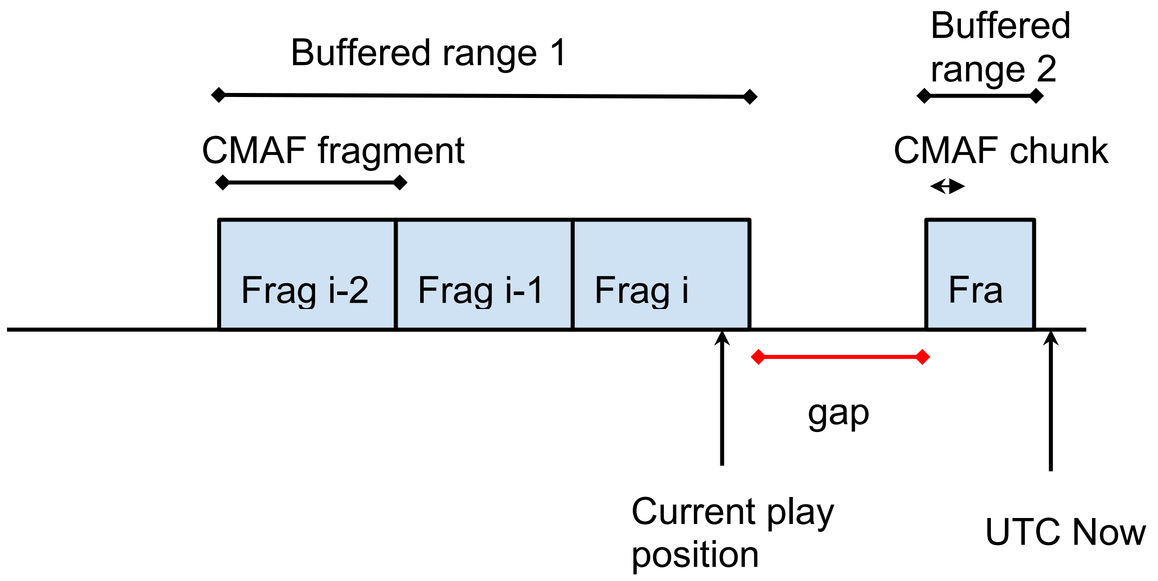
8.20 Low-Latency" Playback over Gaps

8.20.1 Background

Playback over gaps refers to the case that a CMAF/WAVE track is played in low latency mode and the media buffer contains gaps. A client that is playing in low latency mode typically maintains a very small buffer as the current play position is very close to the live edge (UTC now time). By using CMAF chunks, the client can access parts of the fragment prior to the fragment completion. This scenario is depicted in the figure below.



In some cases, unaligned periods, wrong timestamp offsets, false media presentation timestamps or simply missing fragments or chunks can lead to a gap in the media timeline. This leads to a scenario in which the buffer is not continuous. In the Figure below CF i+1 is missing which leads to a gap between CF i and and CF i+2.



MSE implementations do not handle such gaps natively. Instead, playback will stall a few milliseconds before the gap. Note that the concrete point at which the playback will stall depends on the platform/browser implementation. To continue playback, players are required to manually jump over such gaps by seeking to the start of the next buffered range object. In addition, player Implementations need to consider the type of content.

For VoD content, the seek can be triggered immediately. For live content, the player needs to maintain a consistent live edge. Consequently, the player shall wait for the duration of the gap before performing the seek.

8.20.2 Pre-condition

A CMAF Track is available for playback following the properties in clause 5.3.2.4.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5 including the relevant output. This includes:

1) Appending the CMAF Header for the track.
2) Establishing a proper output environment.

8.20.3 Parameters and Variants

The playback has the following parameters:

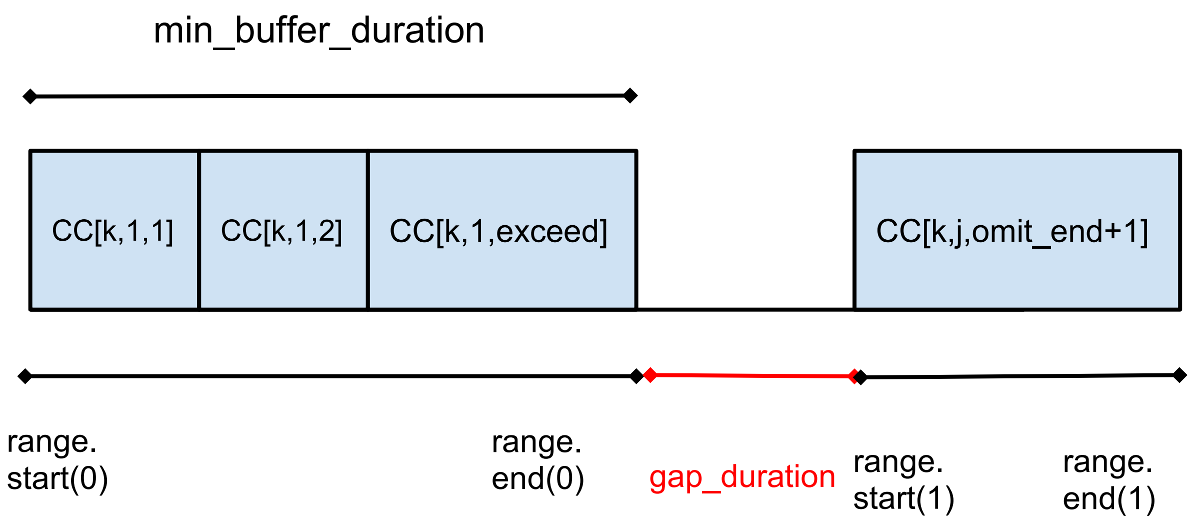
- `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback. This value shall be equal or smaller than tf[k,i+1] - tf[k,i] *(1x fragment duration)* and equal or larger than max(dc[k,i,1],500ms) to simulate low latency playback. Note: On certain platforms a minimum buffer duration is required to trigger the playback. Consequently, the minimum buffer duration shall not be smaller than 500ms. `min_buffer_duration` shall be a multiple of the chunk duration i.e it shall align with a chunk boundary.
- `max_buffer_duration`: Expresses the maximum buffer duration. This value should be larger than `min_buffer_duration` + `dc[k,i,j]` and smaller than `min_buffer_duration` + `df[k,i]`. Default shall be set to `min_buffer_duration` + ½ * `df[k,i]`.
- gap_duration: Expresses the duration of the gap in milliseconds. Default should be set to the duration of a fragment. This value shall not be smaller than the duration of a single CMAF chunk and should be specified as a multiple of the chunk duration.
- playback_mode: Defines the mode of playback which is either "live" or "vod". In vod mode the player immediately seeks over gaps while in live mode the waiting time before the seek is equal to the duration of the gap (to keep a consistent live edge).
- stall_tolerance_margin: Defines a tolerance margin for platforms that possibly take different amounts of time to throw the stall event when reaching the gap.

8.20.4 Stimulus

For a track buffer that supports a media profile, playback over gaps of a CMAF Track `k`, consisting of a sequence of CMAF Header and CMAF Chunks refers to the following actions:

- Append the CMAF Header `CH[k]` to the Source Buffer.
- Set `SourceBuffer.timestampOffset` to `-tf[k,1]`.
- Append CMAF Chunk `CC[k,i,j]` in order starting from `i=1`, and `j=1`, incrementing `j` first to the end and then incrementing `i` and resetting `j=1`, and so on.
- Load as many CMAF Chunks `CC[k,1,j] j = 1...m`, starting from the first Chunk of the track such that the buffer duration is at least `min_buffer_duration` and not larger than or equal to df[k,1]. According to the definition of `min_buffer_duration` the last chunk to be appended is at maximum: `CC[k,1,m]`. In the following we refer to the last chunk that was appended to CC[k,1,exceed].

- Drop as many chunks as required to create a gap with a duration equal to *gap_duration* starting from `CC[k,1,exceed + 1]`. In the following we refer to the last chunk that was omitted as `CC[k,j,omit_end]`
- Load as many CMAF Chunks starting from `CC[k,j,omit_end + 1]` of the track such that the buffer duration reaches `max_buffer_duration`.
- Confirm that two ranges in the `SourceBuffer` exist with:
  - `range.start(0) = 0`
  - `range.end(0) = min(min_buffer_duration,df[k,i=1])`
  - `range.start(1) = range.end(0) + gap_duration`
  - `range.end(1) = max_buffer_duration`



- Initiate playback on the media source and observe:
  - While it is not the last fragment, repeat:
    - As soon as the buffer is below `max_buffer_duration`, append next Chunk CC[k,i,j].
  - Playback starts and continues until `range.end(0)` is reached.
  - Playback stalls.
- Perform a seek to `range.start(1)`
  - `playback_mode = vod`: Seek immediately.
  - `playback_mode = live`: Set a timeout equal to `gap_duration` before seeking.
- Observe:
  - Playback should continue from `range.start(1).`
- Stop at the end of the last Chunk of track in the buffer.

8.20.5 Required Observations

8.20.5.1 General

1) Every sample contained in CMAF Chunks `CC[k,1,1]` to `(CC[k,1,exceed] +/- stall_tolerance_margin)`, and every sample contained in CMAF Chunks `CC[k,1,omit_end+1]` to end of presentation, in increasing presentation time order.
2) The playback duration shall match the duration of the CMAF Track, i.e., `TR[k,S] = TR[k,1] + td[k]` taking into account:
    - Missing starting and ending frames.
    - Potential start frames being rendered before playback.
    - Frozen last frame after the playback ended.
    - The duration of the gap, only for "vod" mode.
    - The stall tolerance margin *stall_tolerance_margin.*

8.20.5.2 Video

If the track is a video track, then the following additional observations are expected: See section 8.2.5.2.

8.20.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

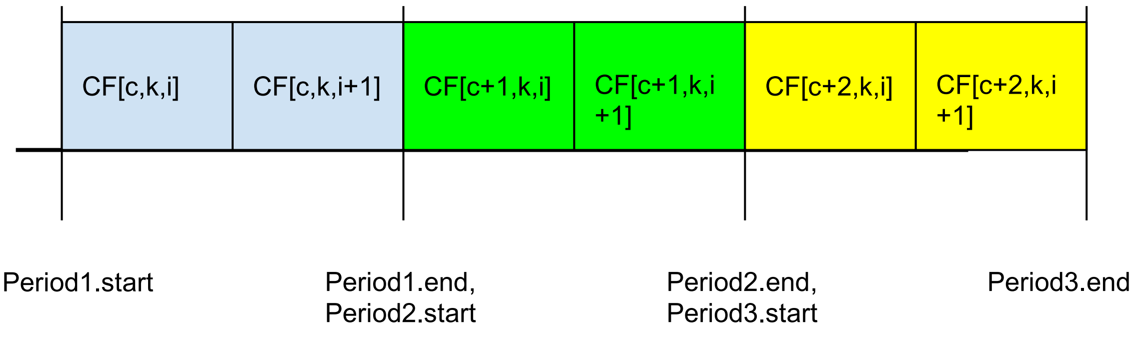
8.21 MSE – AppendWindow

8.21.1 Background

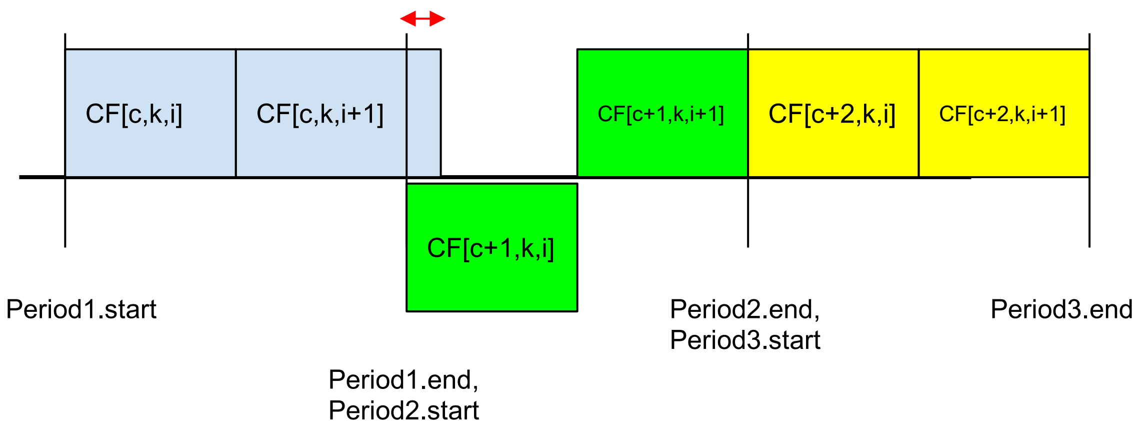
MSE based player implementations use the append window attributes to filter out coded frames while appending. The append window represents a single continuous time range with a single start time and end time. Coded frames with presentation timestamp within this range are allowed to be appended to the `SourceBuffer` while coded frames outside this range are filtered out. The append window start and end times are controlled by the `appendWindowStart` and `appendWindowEnd` attributes respectively `SourceBuffer.appendWindowStart` and `SourceBuffer.appendWindowEnd`.

As an example, a DASH MPD might define multiple periods. In order to avoid appending coded frames that are outside of its period, append windows are used.

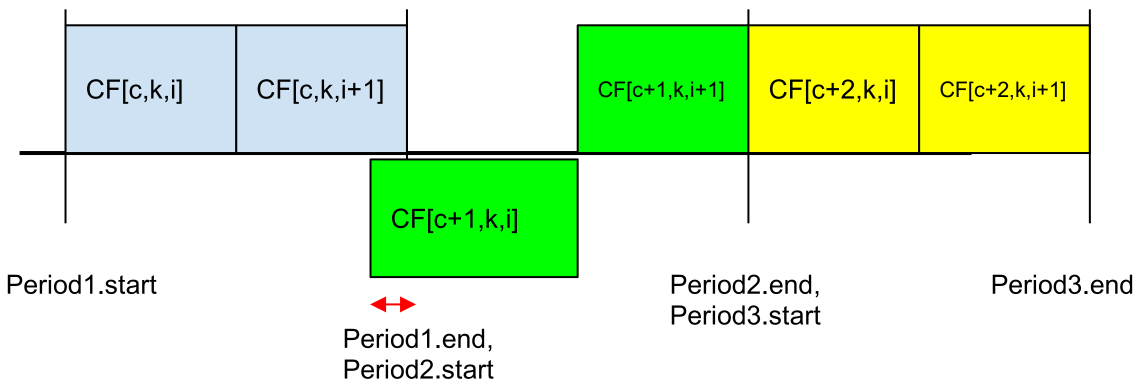
Perfectly aligned fragments



Overlapping fragment at the end of period 1



Overlapping fragment at the beginning of period 2



8.21.2 Pre-Condition

A CMAF Track is available for playback following the properties in clause 5.3.2.4.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5, including the relevant output. This includes:

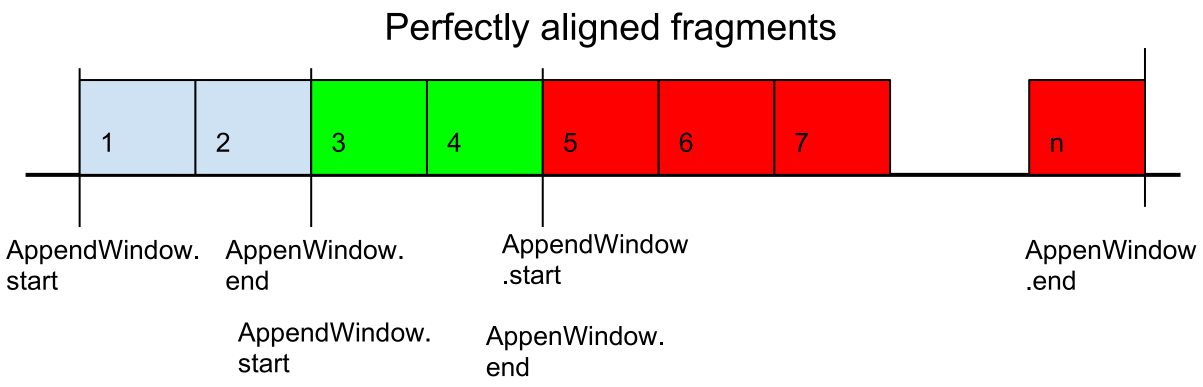
1. Appending the CMAF Header for the track.
2. Establishing a proper output environment.

The CMAF fragments of the required CMAF Switching Sets and CMAF Switching Track are aligned and continuous (compare to perfectly aligned fragments above).

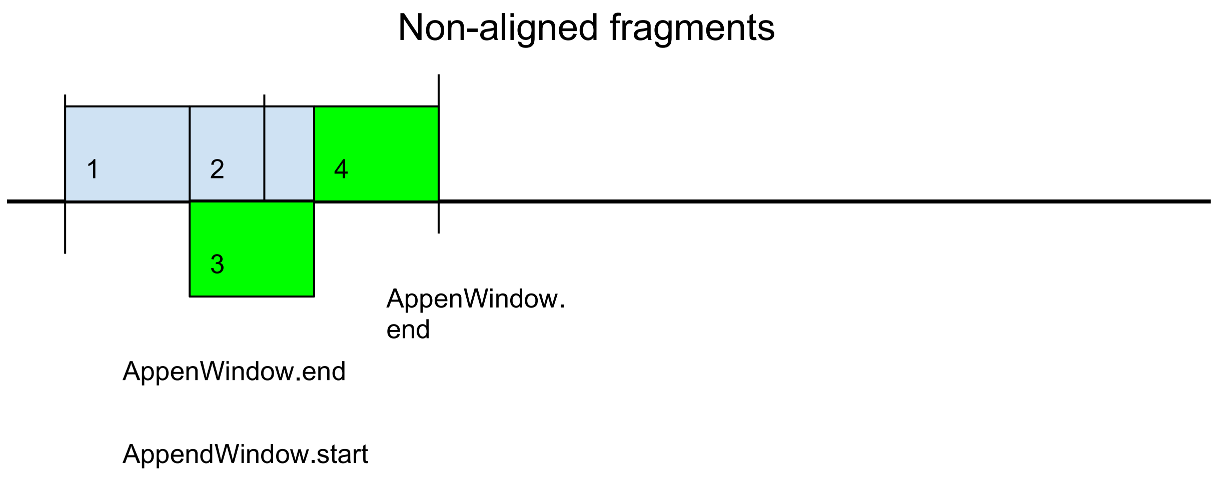
8.21.3 Parameters and Variants

The playback has the following parameters:

- `append_window_boundaries`: The start and end times of the append windows to be created from the CMAF fragments of the CMAF track. The values are depicted in milliseconds. If the test executor wants to avoid overlapping fragments at append window boundaries *start* and *end* need to be aligned with the fragment boundaries. For instance, a fragment duration of 2000ms will result in the following append window boundaries: `[{start:0, end:4000, lastPlayoutEntry:2}, {start: 4000, end: 8000, lastPlayoutEntry:4}}, {start: 8000, end: (n-1) * 2000, lastPlayoutEntry:n}]`.



An example of fragments overlapping the append window boundaries: `[{start:0, end:1500, lastPlayoutEntry:2}}, {start: 2000, end: 3500, lastPlayoutEntry:4}]`. Please note that the `presentationTimeOffset` for fragment x needs to be set accordingly in the playout array if the segments are aligned according to their earliest presentation time.



- `timestamp_offsets[i]`: Provides timestamp offsets for every fragment with the playout position i=1,...,N that is be played out. The timestamp offset is added to the earliest presentation time of the fragment using `SourceBuffer.timestampOffset` (see stimulus below). Note that the `presentationTimeOffset` is added in addition to the "common" calculation of the `SourceBuffer.timestampOffset` (setting `SourceBuffer.timestampOffset` to -`tf[k,1]`).

8.21.4 Stimulus

- Append the CMAF Header `CH[k]` to the Source Buffer.
- Set `SourceBuffer.timeOffset` to -`tf[k,1]`.
- For each entry awb[i] in `append_window_boundaries`:
    - Set `SourceBuffer.timestampOffset` to -`tf[k,1]` + playout[awb[i-1].lastPlayoutEntry + 1].`presentationTimeOffset`.
    - Set `appendWindowStart` to awb[i].start.
    - Set `appendWindowEnd` to awb[i].end.
    - Append CMAF Fragments in *playout* starting from `playout[awb[i-1].lastPlayoutEntry +1].fragmentNumber` and ending at `playout[awb[i].lastPlayoutEntry].fragmentNumber`.
- Initiate play-back on the media source and observe:
    - No frames should be missing.
    - Playback should not stall due to gaps between the segments.

8.21.5 Required Observations

8.21.5.1 General

1) The playback duration shall match the duration of the CMAF Track, i.e., `TR[k,S]` = `TR[k,1]` + `td[k]` taking into account:

- Missing starting and ending frames.
- Potential start frames being rendered before playback.
- Frozen last frame after the playback ended.

8.21.5.2 Video

If the track is a video track, then the following additional observations are expected: See section 8.2.5.2.

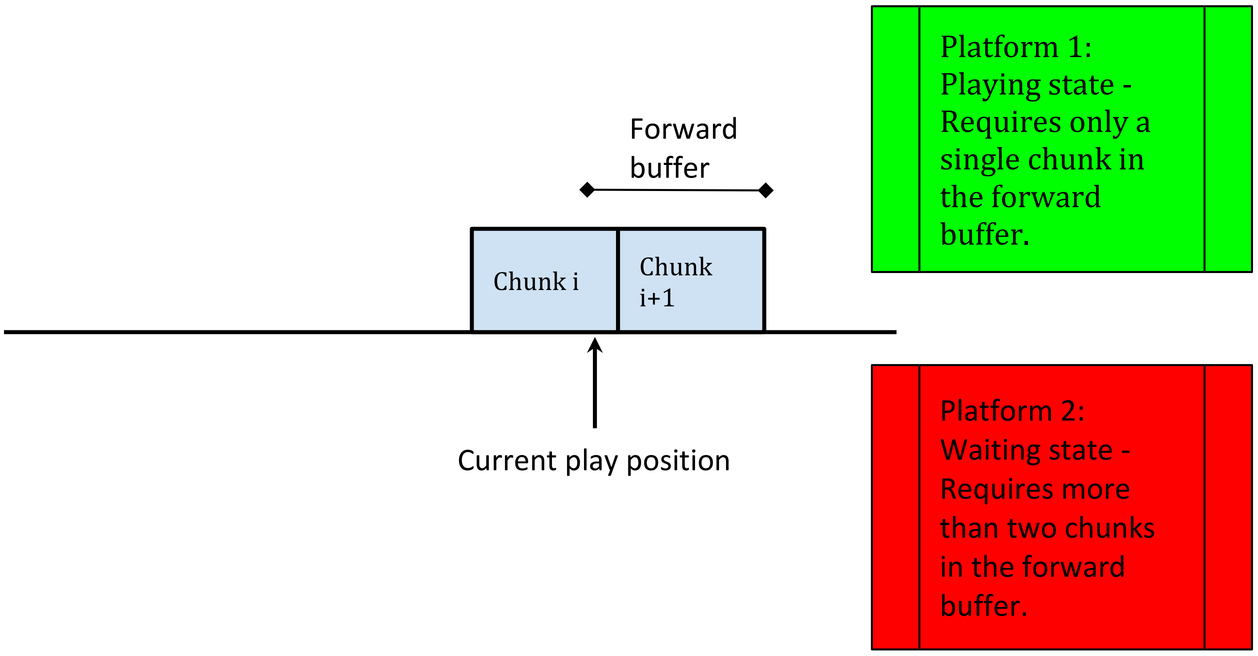
8.21.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

8.22 Low-Latency (2): Short Buffer Playback

8.22.1 Background

Low latency playback is typically done based on very short buffer durations. Depending on the concrete target latency, only a few chunks are in the buffer. MSE capable platforms handle small media buffers differently and consequently stall at different buffer levels. Such differences are illustrated in the figure below. In this case, the first platform requires only a single chunk in the forward buffer for playback. The second platform requires more than two chunks in the forward buffer. Consequently, platform two is in waiting state, although there is still some media data in the buffer to be played.


Forward buffer
Chunk i
Chunk i+1
Current play position
Platform 1: Playing state - Requires only a single chunk in the forward buffer.
Platform 2: Waiting state - Requires more than two chunks in the forward buffer.

8.22.2 Pre-Condition

A CMAF Track is available for playback following the properties in clause 5.3.2.4.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5 including the relevant output. This includes:

1. Appending the CMAF Header for the track.
2. Establishing a proper output environment.

8.22.3 Parameters and Variants

- `maximum_forward_buffer`: Expresses the maximum forward buffer that shall be sufficient for the video element to transition into playing state. This value shall be defined in milliseconds.
- `waiting_timeout`: Expresses the maximum time to wait for the video element to transition into playing state after the buffer has been filled. This value shall be provided in milliseconds.

8.22.4 Stimulus

- Append the CMAF Header `CH[k]` to the Source Buffer.
- Set `SourceBuffer.timestampOffset` to -*`tf[k,1]`*.
- Load and append the CMAF chunks until the buffer reaches a duration equal to or higher than `maximum_forward_buffer`.
- Start a timeout with a duration equal to *`waiting_duration`*.
- Observe:
  - If the video element transitions into playing state, cancel the timeout.
  - If the timeout elapses before the video element has transitioned out of the waiting state, throw an error and terminate the test.
- In case the timeout resolves successfully: Append the rest of the CMAF chunks of CMAF track k.

8.22.5 Required Observations

8.22.5.1 General

1) The playback duration shall match the duration of the CMAF Track, i.e. `TR[k,S]` = `TR[k,1]` + `td[k]` taking into account:
   - Missing starting and ending frames.
   - Potential start frames being rendered before playback.
   - Frozen last frame after the playback ended.
   - The `waiting_duration`.

8.22.5.2 Video

If the track is a video track, then the following additional observations are expected: See section 8.2.5.2.

8.22.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

8.23 Random access from one place in a stream to a different place in the same stream

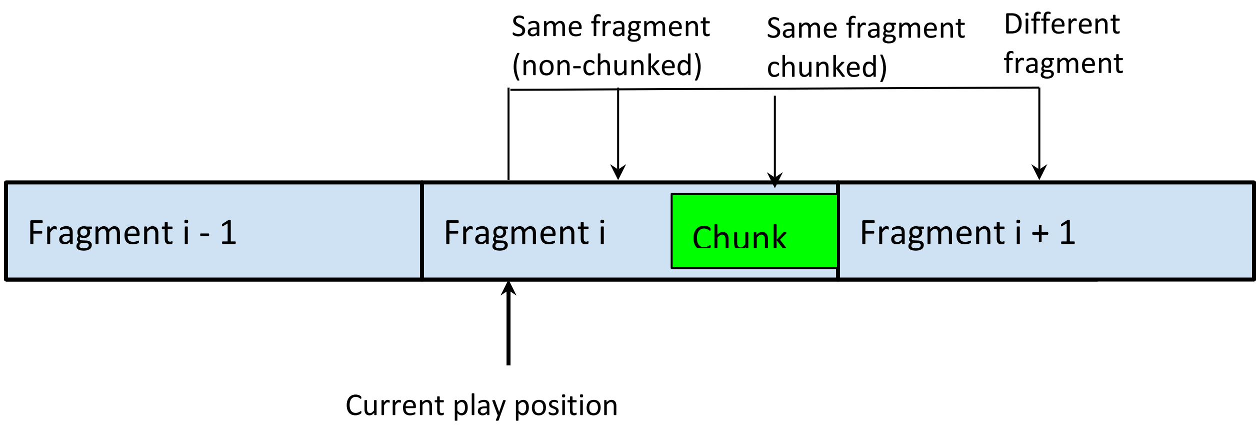
8.23.1 Background

When playing in low latency mode, media players often attempt to maintain a consistent live edge. On platforms on which playback at non-integer speeds >1.0 doesn't work then the alternative is to seek forwards in the stream.

This test covers seeking scenarios where the stream is already playing before the seek is performed. Among other things, this will have different timing considerations and ideally video/audio decoders would not need to be re-initialized.

Different variations are supported:

- Seek into a different CMAF fragment.
- Seek within the same CMAF fragment (not chunked content).
- Seek within the same CMAF fragment (chunked content, different chunk).



8.23.2 Pre-Condition

A CMAF Track is available for playback following the properties in clause 5.3.2.4.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.

A Source Buffer is created as defined in clause 6.5, including the relevant output. This includes:

1) Appending the CMAF Header for the track.
2) Establishing a proper output environment.

8.23.3 Parameters and Variants

- min_buffer_duration: Expresses the initial duration of the SourceBuffer. Once this duration is reached the playback shall be triggered (call to video.play()). This value shall be larger than random_access_from.
- random_access_to: Expresses the playback time to seek to. This value shall be smaller than the total CMAF track duration and shall be defined in milliseconds.
- random_access_from: Expresses the playback time at which the seek is to be done. This value shall be smaller than *random_access_to* and shall be defined in milliseconds.
- random_access_from_tolerance: Defines a tolerance margin at around "random_access_to" the playback time to seek to. The seek may not be triggered in a frame accurate manner.

8.23.4 Stimulus

- Append the CMAF Header *CH[k]* to the Source Buffer.
- Set SourceBuffer.timestampOffset to -*tf[k,1]*.
- Load and append as many CMAF fragments *CF[k,i]* starting from fragment 1 such that the buffer duration is equal or larger than min_buffer_duration.
- Initiate play-back on the media source and observe:
  - Playback should start.
- When the current play position reaches *random_access_from* initialize a seek to the value defined in *random_access_to*.
- Observe.
  - Playback should continue from the presentation time defined in *random_access_to.*
- While it is not the last fragment:
  - As soon as the buffer is min_buffer_duration or below, append next fragment CF[k,i].
- Stop the playback at the end of the last fragment in buffer.

8.23.5 Required Observations

8.23.5.1 General

1) The playback duration shall match the duration of the CMAF Track, i.e. TR[k,S] = TR[k,1] + td[k] taking into account:
   - Missing starting and ending frames.
   - Potential start frames being rendered before playback.
   - Frozen last frame after the playback ended.
   - Skipped frames between *random_access_from* and *random_access_to.*

8.23.5.2 Video

If the track is a video track, then the following additional observations are expected:

1) The presented sample shall match the one reported by the `currentTime` value within the tolerance of `+/- (2/framerate + 20ms).`
2) Every video frame shall be rendered from fragment 1 till *random_access_from* +/- random_access_from_tolerance and from *random_access_to* till the end of last fragment. And the video frames shall be rendered in increasing presentation time order.
3) The start-up delay should be sufficiently low. See section 8.2.5.2.

8.23.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

9 WAVE CONTENT PLAYBACK REQUIREMENTS

9.1 Introduction

This clause deals with requirements of playback of WAVE programs.

9.2 Regular Playback of a CMAF Presentation

9.2.1 Background

This case refers to the case for which a CMAF Presentation is provided and is played back. The playback instructs the player to play multiple tracks, but at most one of each media type.

9.2.2 Pre-condition

A WAVE Presentation is available for playback following the properties in clause 5.3.3. The WAVE Presentation has at least a Switching Set of type video and one of type audio. It may also have subtitles. The earliest presentation time of the first CMAF Fragment in each track are identical and are 0. The CMAF Presentation has a duration.

A Media Source is established as defined in clause 6.2.

For each media type:

1) The capability discovery as defined in clause 6.3 using the parameters assigned to the track was successful.
2) A Source Buffer is created as defined in clause 6.4.

9.2.3 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback <needs >.

9.2.4 Stimulus

For each source buffer an independent process is run.

Start play is done on the media source/element ➔ observe T1.

The first rendered sample is specific to the source buffer T2[s] with s source buffer index.

Stop play is done on the media source/element.

9.2.5 Required Observations

9.2.5.1 General

1) The playback duration shall match the duration of the CMAF Track, i.e., `TR[k,S] = TR[k,1] + td[k]` taking into account:
    - Missing starting and ending frames.
    - Potential start frames being rendered before playback.
    - Frozen last frame after the playback ended.
    - Audio duration should match video duration.
2) The WAVE presentation starts with the earliest video and audio sample that corresponds to the same presentation time as the earliest video sample.

9.2.5.2 Video

If the track is a video track, then the following additional observations are expected: See section 8.2.5.2.

9.2.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

9.2.5.4 Audio-Video Synchronization

For audio-video synchronization in this requirements group, and for other requirements groups that follow, [ITU SYNC] may be consulted for further information. [ITU SYNC] at Appendix 1 indicates, *"Tests conducted have shown that the thresholds of detectability are about + 45 ms to –125 ms… "*

It is preferred to keep to a measurement resolution of `20 ms`, thus, the quantitative requirements in this specification are kept to multiples of `20 ms`, such as `"+40 ms"`.

The following is the requirement for audio-video synchronization:

The `mediaTime` of the presented audio sample shall match the one reported by the video `currentTime` value within the tolerance of `+40 ms / -120 ms`.

9.3 Random Access of a WAVE Presentation

9.3.1 Background

In scenarios such as live programs, the client accesses a Presentation while ongoing to join at the live event. This is a typical case that media needs to be accessed and played back at a random access time.

9.3.2 Pre-condition

A WAVE Presentation is available for playback following the properties in clause 5.3.3. The WAVE Presentation has at least a Switching Set (likely only one track) of type video and one of type audio. There may also be subtitles. The earliest presentation time of the first CMAF Fragment in each track are identical and are 0. The CMAF Presentation has a duration.

A Media Source is established as defined in clause 6.2.

For each media type:

1) The capability discovery as defined in clause 6.3 using the parameters assigned to the track was successful.
2) A Source Buffer is created as defined in clause 6.4.

9.3.3 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback <needs >.
2) `random_access_fragment`: The CMAF Fragment number of the video track to start with.

9.3.4 Stimulus

For each source buffer an independent process is run.

The `td[random_access_fragment]` is the decode time of the first video CMAF Fragment.

The first video CMAF Fragment to be appended is `CF[random_access_fragment]`.

Pick the CMAF Fragment with the largest index that has a presentation time that is smaller than or equal to the earliest presentation time of the CMAF Video fragment `CF[random_access_fragment]` .

Set the presentation time offset to `td[random_access_fragment]`.

Start play on Media Source Element.

9.3.5 Required Observations

9.3.5.1 General

1) The playback duration shall match the duration of the CMAF Track – i.e., `TR[k,S] = TR [k, s1] + td[k] - tf[k,i= random_access_fragment]`, taking into account:
    - Missing starting and ending frames.
    - Potential start frames being rendered before playback.
    - Frozen last frame after the playback ended.
    - Audio duration should match video duration.
2) The video presentation starts with the earliest video sample of `random_access_fragment` and audio sample that corresponds to the same presentation time as the earliest video sample.

9.3.5.2 Video

If the track is a video track, then the following additional observations are expected: See section 8.2.5.2.

9.3.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

9.3.5.4 Audio-Video Synchronization

The `mediaTime` of the presented audio sample shall match the one reported by the video `currentTime` value within the tolerance of `+40 ms / -120 ms`.

9.4 Splicing of WAVE Program with Baseline Constraints

9.4.1 Background

Apps want to splice to content that was properly conditioned based on the WAVE Program Baseline Constraints. The expectation is that the device plays back these splices seamlessly.

9.4.2 Pre-condition

Four CMAF Switching Sets are available:

- Two CMAF Switching Sets for video and two Switching Sets for audio are available.
- Each pair of Switching Sets follows the properties in clause 5.3.3.5. Consequently, all CMAF tracks in both Switching Sets conform to one media profile, one CMAF Principal Header exists for initialization of playback and this CMAF Principal Header can be appended to the source buffer without triggering a reinitialization of the decoding and rendering platform.
- In the first switching set pair, there exists at least one media time greater than 0, for which the Audio and Video Switching Set coincides with a Fragment Boundary.

A Media Source is established as defined in clause 6.2.

The capability discovery as defined in clause 6.4 using the parameters assigned to the tracks was successful using the CMAF Principal Header parameters.

An audio and a video source Buffer is created as defined in clause 6.5 including the relevant outputs. This includes:

- Appending the CMAF Principal Header for each of the media type.
- Establishing a proper output environment.

9.4.3 Parameters and Variants

The playback has the following parameters:

- `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback.

- An aligned media time `Tsplice` in the first Audio and Video Switching Set that coincide with a Fragment Boundary and is instructed for playback.

9.4.4 Stimulus

For a track buffer that supports a media profile, playback of two WAVE Baseline Splice Constraints Switching Sets of a CMAF Switching Set, consisting of K tracks, the following applies:

- Set presentation time offset to `0`.
- For each track from the first presentation
    - Set the `LastHeader` to the CMAF Principal Header `CH*` used for initialization.
    - Load as many CMAF fragments `CF[k,i]` starting from fragment 1 such that the buffer duration is at least `min_buffer_duration` using the Append-Algorithm-2, outlined below.
    - Once reached the `min_buffer_duration`, initiate play-back on the media source and observe:
        - The measured time when playback is initiated is `Ti`.
        - The measured time when the first sample is rendered at time `TR[s=1]`.
        - The measured time when sample s is rendered is time `TR[s]`.
- Once all fragments are loaded up to time `TSplice`, set presentation time offset to `0`.
- For each track from the second presentation:
    - If the `LastHeader` is not equal to the CMAF Principal Header `CH*` of the track, append CMAF header.
    - Load as many CMAF fragments `CF[k,i]` starting from fragment 1 using the Append-Algorithm-2, outlined below.
    - While it is not the last fragment:
        - As soon as the `buffer` is equal to `min_buffer_duration` or below, append next fragment `CF[k,i]` using Append-Algorithm-2, outlined below.
- Once all fragments are loaded up to the end of the second presentation, set presentation time offset to `TSplice`.
- For each track from the second presentation:
    - If the `LastHeader` is not equal to the CMAF Principal Header `CH*` of the track, append CMAF header.
    - Load as many CMAF fragments `CF[k,i]` starting from fragment with media `Tsplice` using the Append-Algorithm-2, outlined below.
    - While it is not the last fragment:
        - As soon as the `buffer` is equal to `min_buffer_duration` or below, append next fragment `CF[k,i]` using Append-Algorithm-2, outlined below.
- Stop at the end of the last CMAF Fragment in the buffer.

Append-Algorithm-2:

- For each CMAF Fragment position `i=1,…,N`:
    - If CMAF Header `CH[k = playout[i]]!= LastHeader`:
        - Append the CMAF Header `CH[k=playout[i]]` to the Source Buffer.
        - Set `LastHeader` to `CH[k=playout[i]]`.
    - Append CMAF Fragment `CF[k=playout[i],i]`.

9.4.5 Required Observations

9.4.5.1 General

1) The playback duration shall match the duration of the CMAF fragments in both CMAF Presentations, taking into account:
    - Missing starting and ending frames.
    - Potential start frames being rendered before playback.
    - Frozen last frame after the playback ended.
    - Audio duration should match video duration.
2) The video presentation starts with the earliest video sample and audio sample that corresponds to the same presentation time as the earliest video sample.
3) At splice point 1, when the second presentation starts, the presentation is observed continuous without gap in playback within the tolerance of +/- (2/framerate + 20ms).
4) At splice point 2, when there is a return to the first presentation, no gap is observed in playback.

9.4.5.2 Video

If the track is a video track, then the following additional observations are expected: See section 8.2.5.2.

9.4.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

9.4.5.4 Audio-Video Synchronization

The `mediaTime` of the presented audio sample shall match the one reported by the video `currentTime` value within the tolerance of +40 ms / -120 ms.

9.5 Joint Playback of Video and Subtitles

9.5.1 Background

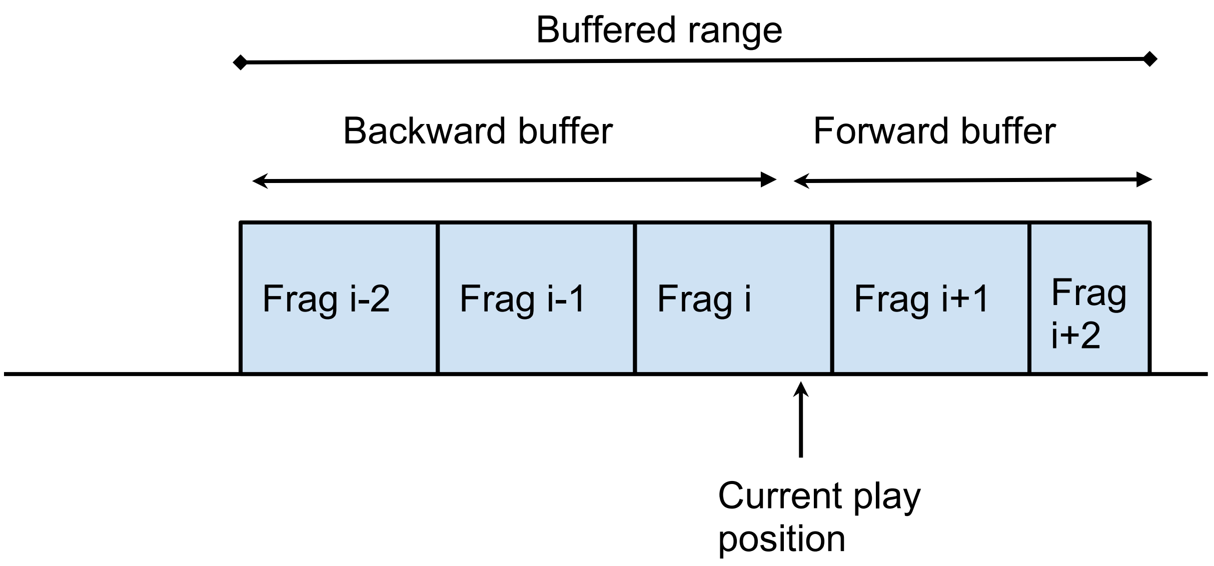
This test specifies the joint playback of video and subtitles and the proper rendering.

No tests are defined for this feature in the first version of the specification.

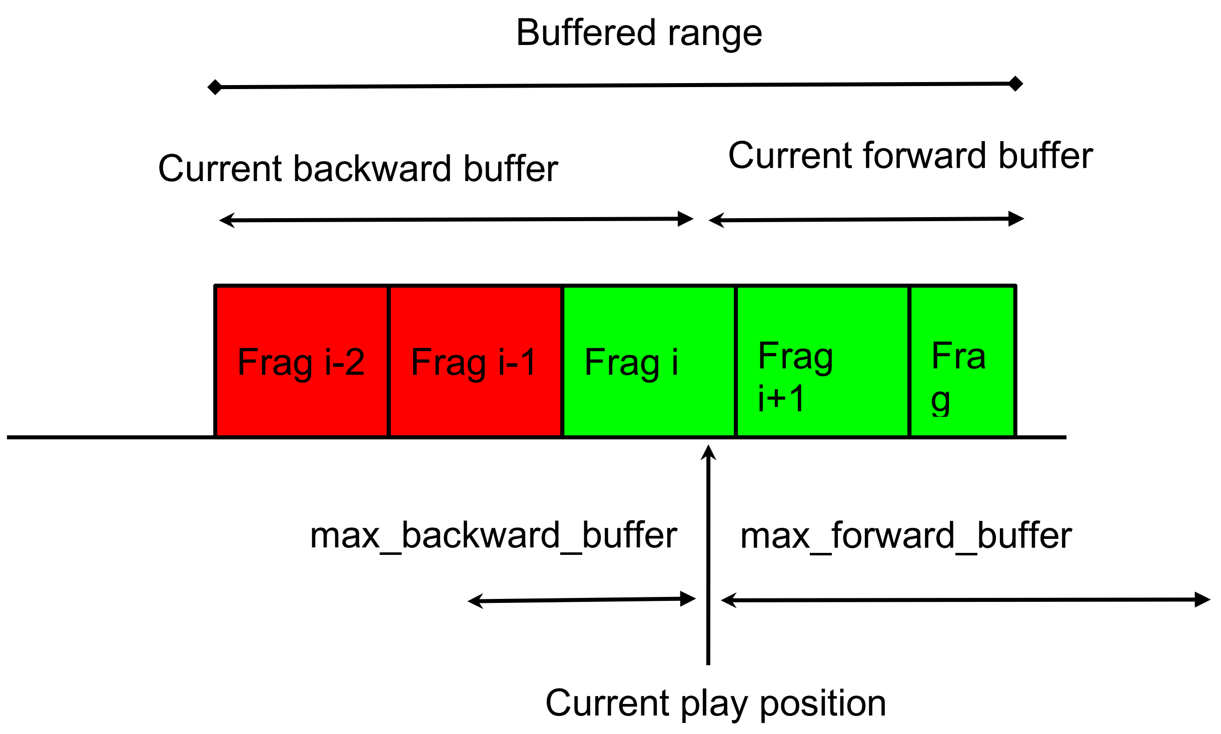
9.6 Long Duration Playback of CMAF Presentations

9.6.1 Background

This requirement addresses the proper playback of a CMAF track over a longer time, in particular no drift in the playout occurs. During the playback, the client maintains a forward and a backward buffer as depicted below:



An important aspect of long duration playback is proper buffer management. The number of fragments that an MSE capable platform can keep in its buffer depends on various factors such as the general platform capabilities and the bitrate and duration of the fragments. Consequently, a buffer cleanup logic in the application is required to avoid a full buffer and *QuotaExceededError* exceptions thrown by the MSE.



9.6.2 Pre-Condition

A WAVE Presentation is available for playback following the properties in clause 5.3.3. The WAVE Presentation has at least a Switching Set (likely only one track) of type video and one of type audio. The Presentation may also have subtitles. The earliest presentation time of the first CMAF Fragment in each track are identical and are 0. The CMAF Presentation has a duration.

A Media Source is established as defined in clause 6.2.

For each media type:

- The capability discovery as defined in clause 6.4 using the parameters assigned to the track was successful.
- A Source Buffer is created as defined in clause 6.5.
    - Appending the CMAF Header for the track.
    - Establishing a proper output environment.

9.6.3 Parameters and Variants

- `min_buffer_duration`: Expresses the initial duration of the `SourceBuffers`. Once this duration is reached the playback shall be triggered (call to video.play()).
- `max_backward_buffer`: Expresses the maximum backward buffer in seconds.
- `playback_duration`: Expresses the target playback duration in seconds. If no `playback_duration` is given, all CMAF fragments in the CMAFs track shall be presented. The `playback_duration` shall not exceed the total duration of all CMAF fragments in the CMAF tracks.

9.6.4 Stimulus

- For each SourceBuffer:
  - Append the CMAF Header *CH[k]* to the Source Buffer.
  - Set SourceBuffer.timestampOffset to -*tf[k,1]*.
  - Load and append as many CMAF fragments *CF[k,i]* starting from fragment 1 such that the buffer duration is equal or larger than min_buffer_duration.
- Once reached the min_buffer_duration on all the SourceBuffers, initiate playback on the media source and observe:
  - While it is not the last fragment:
    - As soon as the buffer is equal to min_buffer_duration or below, append the next fragment CF[k,i].
  - After each append operation of a CMAF fragment, check:
    - If the current backward buffer exceeds the value of *max_backward_buffer*. If true, remove all the data in the buffer up to the start of the fragment that is within the threshold of *max_backward_buffer.* Compare to the figure above: *Frag i-2* and *Frag i-1* are removed because they exceed the value of *max_backward_buffer*.
  - Stop once the total duration of all appended CMAF fragments is equal or higher than playback_duration. If no playback_duration is given, stop at the end of the last CMAF Fragment in the buffer.

9.6.5 Required Observations

9.6.5.1 General

1) The playback duration shall match the duration of the CMAF Track, i.e., TR[k,S] = TR[k,1] + td[k] taking into account:
   - Missing starting and ending frames.
   - Potential start frames being rendered before playback.
   - Frozen last frame after the playback ended.
   - Audio duration should match video duration.
2) The video presentation starts with the earliest video sample and audio sample that corresponds to the same presentation time as the earliest video sample.

9.6.5.2 Video

If the track is a video track, then the following additional observations are expected: See section 8.2.5.2.

9.6.5.3 Audio

If the track is an audio track, then the additional observations as per clause 8.2.5.3 are expected.

9.6.5.4 Audio-Video Synchronization

The mediaTime of the presented audio sample shall match the one reported by the video currentTime value within the tolerance of +40 ms / -120 ms.

10 GENERAL CMAF REQUIREMENTS AND TESTS

10.1 Introduction

The WAVE content specification [WAVE-CON] relies on CMAF content specification [CMAF]. It was identified that CMAF itself has several options that are worthwhile to be tested, more or less independent of the codec. In order to address such baseline tests, only media profiles from the CMAF presentation profile as defined in Annex A.1 are of ISO/IEC 23000-19 [CMAF] are used. Specifically:

- The CMFHD presentation profile as defined in Annex A.1.2 of CMAF is intended to provide basic interoperability of unprotected content on the widest range of Internet video devices. For this, the media profiles included are:
    - For video, the `'cfhd'` media profile is included.
    - For audio, the `'caac'` media profile is included.
    - For subtitle, the `'im1t'` media profile is included.
- Annex A.1.3 and Annex A.1.4 define presentation profiles for `'cenc'` and `'cbcs'` encrypted content, respectively.

Specific test vectors and playback tests for media profiles are documented in clauses 11, 12 and 13.

10.2 Video

10.2.1 General

For video, general playback requirements are tested with the media profile CMAF AVC HD as defined in [WAVE-CON], clause 4.2 and [CMAF].

Capability Discovery and Source Buffer Initialization for this media profile are addressed in clauses 11.2.2 and 11.2.3, respectively.

10.2.2 Source Content Options

Content that conforms to a particular WAVE Media Profile may have any resolution within the limits specified for that Media Profile.

Devices that support one or more Media Profiles that have a defined maximum resolution of 1920x1080 shall support the decoding and display of pictures with the resolutions listed below for those Media Profiles. This does not preclude the use of other resolutions in WAVE content. However, a limited number of resolutions are listed here to ease device testing.

Horizontal	Vertical
1920	1080
1600	900
1280	720
1024	576
960	540
852	480

Horizontal	Vertical
768	432
720	404
704	396
640	360
512	288
480	270
384	216
320	180
192	108

Devices that support one or more Media Profiles that have a defined maximum resolution of 3840x2160 shall support the decoding and display of pictures with the resolutions listed below for those Media Profiles. This does not preclude the use of other resolutions in WAVE content. However, a limited number of resolutions are listed here to ease device testing.

Horizontal	Vertical
3840	2160
3200	1800
2560	1440
1920	1080
1600	900
1280	720
1024	576
960	540
852	480
768	432
720	404
704	396
640	360
512	288
480	270
384	216
320	180
192	108

Devices that support one or more Media Profiles that have a defined maximum frame rate of 60 Hz shall support the decoding and display of pictures with the following frame rates for those Media Profiles. This does not preclude the use of other frame rates in WAVE content. However, a limited number of frame rates are listed here to ease device testing:

- 6/1.001 Hz, 12/1.001 Hz and 24/1.001 Hz
- 6 Hz, 12 Hz, 24 Hz
- 6.25 Hz, 12.5 Hz, 25 Hz and 50 Hz

- 7.5/1.001 Hz, 15/1.001 Hz, 30/1.001 Hz and 60/1.001 Hz
- 7.5 Hz, 15 Hz, 30 Hz and 60 Hz

Devices shall support switching frame rates within each family listed above.

Devices supporting a particular Media Profile shall support all of the color coding and transfer characteristics options defined for that Media Profile.

10.2.3 Content Options

The following content options are highlighted for the purposes of testing the playback requirements and are considered to be the primary options for this version of the specification. Note that other options supported within the media profile are also valid, but content providers should not expect that they have been tested.

Content options in *italics* are considered of higher priority for testing and are prioritized as they may for example cause problems during playback.

General CMAF Options (see clause 7 of CMAF [CMAF])

- Compositions offsets and Timing
    - `'cmfc'`
        - 7.5.17 Track Run Box (`'trun'`)
            - v0.
        - 7.5.13 Edit List Box (`'elst'`)
            - Absent – SAP type 1
            - Present – SAP type 2 (typically with leading B-Pictures)
        - `default_sample_flags, sample_flags` and `first_sample_flags` neither set in the `TrackFragmentHeaderBox` nor `TrackRunBox`.
    - `'cmfc'` and `'cmf2'`
        - 7.5.17 Track Run Box (`'trun'`)
            - v1 – combined with SAP type 1
            - v1 – combined with SAP type 2 (typically with leading B-Pictures)
        - 7.5.13 Edit List Box (`'elst'`)
            - Absent
        - `default_sample_flags, sample_flags` and `first_sample_flags` set in:
            - `TrackFragmentHeaderBox`
            - `TrackRunBox`
- 7.4.5 Event Message Box (`'emsg'`)
    - *Absent*
    - v0
    - v1

Encrypted Content

- *Unencrypted*

Source Content Options

- 50Hz video
- 60Hz video, 60/1.001 Hz
- *Priority is given to 1920x1080p25 and 1280x720p50.*
- 16:9 picture aspect ratio square pixel [CMAF], clause 9.2.3 and 9.4.2.2.2.
- At least one source sequence with another picture aspect ratio, non-square pixel with conformance cropping following the example in Annex C.8 of [CMAF] specification.

Encoding and Packaging options

- SEI and VUI
    - Without picture timing SEI message.
    - Without VUI timing information.
- Sample entry, see CMAF clause 9.4.1.2. Two options:
    - `'avc1'` *sample entry type (parameter sets within the CMAF Header).*
    - `'avc3'` sample entry (parameters are inband).
- Initialization Constraints.
    - Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4.
- CMAF Fragment durations
    - *2 seconds*
    - 5 seconds
- Fragments containing one or multiple moof/mdat pairs
    - *Fragment is 1 chunk.*
    - *Fragment contains multiple chunks (p-frame to p-frame with b-frames).*
    - Each sample constitutes a chunk (p-frame only).
- Switching Set encoding following CMAF, Annex C.9 recommendation. At least one of each of the following options:
    - No spatial or temporal subsetting.
    - Spatial subsampling of video according to CMAF, Annex C.2 recommendation.
    - Spatial and temporal subsampling and scaling of video according to CMAF, Annex C.1 recommendation for each 50 Hz and 60 Hz family.
    - CMAF, Annex C.9.3 recommendation for video on demand.
    - CMAF, Annex C.9.4 recommendation for low-latency live.

10.2.4 Test Content

The test content for AVC video is documented in Table 2.

Table 2: Test content for AVC video

		Desired Stream Duration	Comment	Test stream								
				1	2	3	10	11	12	13	14	15
Test stream summary				One set of AVC content options	All AVC content options different from stream 1.	Same as 1 except for AVC content options that have 3 values different from 1 and 2.	Same as 1 but with only picture timing SEI message different.	Same as 1 but with only VUI timing different.	Same as 1 but with different sample entry type.	Same as 1 but with 3rd possible sample entry type.	Same as 1 but with only CMAF fragment duration different.	Same as 1 but with only initialization constraints different
Test stream content options	with and without picture timing SEI message.			With	Without	With	Without	With	With	With	With	With
	with and without VUI timing information.			Without	With	Without	Without	With	Without	Without	Without	Without
	Sample entry, see CMAF clause 9.4.1.2			avc1 sample entry type (parameter sets within the CMAF Header)	avc3 sample entry type (in-band parameter sets) without parameter sets within the CMAF header	avc3 sample entry type (in-band parameter sets) with parameter sets within the CMAF header	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc3 sample entry type (in-band parameter sets) without parameter sets within the CMAF header	avc3 sample entry type (in-band parameter sets) with parameter sets within the CMAF header	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)
	CMAF Fragment durations			2	5	2	2	2	2	2	5	2
	Initialization Constraints			Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Single initialization constraints, see CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Single initialization constraints, see CMAF clause 7.3.4.2 and 9.2.11.4
	Fragments containing one or multiple moof/mdat pairs			Fragment is 1 chunk	Fragment contains multiple chunks (p-frame to p-frame with b-frames)	Each sample constitutes a chunk (p-frame only)	Fragment is 1 chunk	Fragment is 1 chunk	Fragment is 1 chunk	Fragment is 1 chunk	Fragment is 1 chunk	Fragment is 1 chunk
	Resolution			1920x1080	1920x1080	1920x1080	1920x1080	1920x1080	1920x1080	1920x1080	1920x1080	1920x1080
	Frame rate (2.0=100/120,1.0=50/59.94/60, 0.5=25/29.97/30, 0.25=12.5/15)			0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
	Bit rate											
	Duration of stream			30s	30s	30s	30s	30s	30s	30s	30s	30s
	AVC profile and level			High 4.0	High 4.0	High 4.0	High 4.0	High 4.0	High 4.0	High 4.0	High 4.0	High 4.0
	CMAF media profile			cfhd	cfhd	cfhd	cfhd	cfhd	cfhd	cfhd	cfhd	cfhd

		16	17	19	20	21	22	23	24	25
Test stream summary		Same as 1 but with only moof/mdat pair different	Same as 1 but with third option for moof/mdat pairs.	Same as 1 but 60s duration for switching set tests	Same as 1 but 1080p50/60 instead of 1080p25/30.	Same as 1 but 1600x900	Same as 1 but 1280x720	Same as 22 but 720p50/60 instead of 720p/25/30.	Same as 1 but 1024x576, higher birrate	Same as 1 but 1024x576 lower bitrate
Test stream content options	with and without picture timing SEI message.	With	With	With	With	With	With	With	With	With
	with and without VUI timing information.	Without	Without	Without	Without	Without	Without	Without	Without	Without
	Sample entry, see CMAF clause 9.4.1.2	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)
	CMAF Fragment durations	2	2	2	2	2	2	2	2	2
	Initialization Constraints	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4
	Fragments containing one or multiple moof/mdat pairs	Fragment contains multiple chunks (p-frame to p-frame with b-frames)	Each sample constitutes a chunk (p-frame only)	Fragment is 1 chunk	Fragment is 1 chunk					
	Resolution	1920x1080	1920x1080	1920x1080	1920x1080	1600x900	1280x720	1280x720	1024x576	1024x576
	Frame rate (2.0=100/120,1.0=50/59.94/60, 0.5=25/29.97/30, 0.25=12.5/15)	0.5	0.5	0.5	1	0.5	0.5	1	0.5	0.5
	Bit rate									
	Duration of stream	30s	30s	60s	60s	60s	60s	60s	60s	60s
	AVC profile and level	High 4.0	High 4.0	High 4.0	High 4.2	High 4.0	High 4.0	High 4.0	High 4.0	High 4.0
	CMAF media profile	cfhd	cfhd	cfhd	chdf	cfhd	cfhd	cfhd	cfhd	cfhd

		26	27	28	29	30	31	32	33	34	splice_main	splice_ad
Test stream summary		Same as 1 but 960x540	Same as 1 but 852x480	Same as 1 but 768x432.	Same as 1 but 720x404	Same as 1 but 704x396	Same as 1 but 640x360	Same as 1 but 512x288	Same as 1 but 480x270	Same as 33 but half frame rate	Same as 1 but ...	Same as 1 but ...
Test stream content options	with and without picture timing SEI message.	With	With	With	With	With	With	With	With	With	With	With
	with and without VUI timing information.	Without	Without	Without	Without	Without	Without	Without	Without	Without	Without	Without
	Sample entry, see CMAF clause 9.4.1.2	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)	avc1 sample entry type (parameter sets within the CMAF Header)
	CMAF Fragment durations	2	2	2	2	2	2	2	2	2	2	2
	Initialization Constraints	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4	Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4
	Fragments containing one or multiple moof/mdat pairs										Fragment is 1 chunk	Fragment is 1 chunk
	Resolution	960x540	852x480	768x432	720x404	704x396	640x360	512x288	480x270	480x270	1920x1080	1920x1080
	Frame rate (2.0=100/120,1.0=50/59.94/60, 0.5=25/29.97/30, 0.25=12.5/15)	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.25	0.5	0.5
	Bit rate											
	Duration of stream	60s	60s	60s	60s	60s	60s	60s	60s	60s		
	AVC profile and level	High 4.0	High 4.0	High 4.0	High 4.0	High 4.0	High 4.0	High 4.0	High 4.0	High 4.0	High 4.0	High 4.0
	CMAF media profile	cfhd	cfhd	cfhd	cfhd	cfhd	cfhd	cfhd	cfhd	cfhd	cfhd	cfhd

For more details, the test code matrix is attached to the specification as 'WAVE test content _ test code sparse Matrix – AVC.xlsx'.

NOTE: The attached file is a dump of the online version of the AVC test matrix at the time of the technical freeze of this version of the specification. The online version will be updated and maintained with bug fixes and additional content.

10.2.5 Playback Requirements

Support for general CMAF video track playback includes:

- The requirement to support the following playback requirements as documented in clause 8:
    - 8.2 Sequential Track Playback for all prioritized content options
    - 8.3 Random Access to Fragment for all prioritized content options
    - 8.4 Random Access to Time for all prioritized content options
    - 8.5 Switching Set Playback for all prioritized content options
    - 8.6 Regular Playback of Chunked Content for all prioritized content options
    - 8.7 Regular Playback of Chunked Content, non-aligned append for all prioritized content options
    - *8.8 Playback over WAVE Baseline Splice Constraints* for all prioritized content options
    - 8.11 Full Screen Playback of Switching Sets for all prioritized content options
    - 8.13 Restricted Splicing of Encrypted Content for all prioritized content options
    - 8.14 Sequential Playback of Encrypted and Non-Encrypted Baseline Content for all prioritized content options
    - 8.15 Source Buffer Re-Initialization (without changeType) for all prioritized content options
    - 8.16 Source Buffer Re-Initialization (with changeType) for all prioritized content options
    - 8.17 Buffer Underrun and Recovery for all prioritized content options
    - 8.18 Truncated Playback and Restart for all prioritized content options
    - 8.19 Low-Latency (1): Initialization for all prioritized content options
    - 8.20 Low-Latency"   Playback over Gaps for all prioritized content options
    - 8.21 MSE – AppendWindow for all prioritized content options

    - 8.22 Low-Latency (2): Short Buffer Playback for all prioritized content options
    - 8.23 Random access from one place in a stream to a different place in the same stream for all prioritized content options
- The recommendation to support the following playback requirements as documented in clause 8:
    - 8.9 Out-Of-Order Loading for all prioritized content options
    - 8.10 Overlapping Fragments for all prioritized content options
    - 8.12 Playback of Encrypted Content for all prioritized encrypted content options
- The option to support all applicable requirements from clause 8 for non-prioritized content options.

10.2.6 Test Cases

The test cases for AVC video are documented in the attached sheet as 'WAVE test content _ test code sparse Matrix - AVC.xlsx'.

> NOTE: The attached file is a dump of the online version of the AVC test matrix at the time of the technical freeze of this version of the specification. The online version will be updated and maintained with bug fixes and additional content.

10.3 Audio

10.3.1 General

For audio, general playback requirements are tested with the media profile CMAF AAC Core as defined in [WAVE-CON], clause 4.3 and [CMAF].

Capability Discovery and Source Buffer Initialization for this media profile are addressed in clause 12.1.2.

10.3.2 Content Options

The following content options are highlighted for the purposes of testing the playback requirements and are considered to be the primary options for this version of the specification. Note that other options supported within the media profile are also valid, but content providers should not expect that they have been tested.

Content options in *italics* are considered of higher priority for testing and are prioritized as they may for example cause problems during playback.

General CMAF Options (see clause 7)

- Compositions offsets and Timing.
- Conformant to `'cmf2'`, branded as `'cmfc'` and `'cmf2'`. (Note: conformant only to `'cmfc'` is covered in clause 10.2.3).
    - 7.5.17 Track Run Box (`'trun'`)
        - v0
        - v1
    - 7.5.13 Edit List Box (`'elst'`)

        - Present with v0
        - Absent with v1

Source Content Options

- Stereo only

Encoding and Packaging options

- CMAF brand: `caac`
- Codec: MPEG-4 AAC-LC (subparameter '`mp4a.40.2`')
- Sampling Frequency only 48kHz
- Channel: Stereo
- Without dynamic range/loudness control.
- program_config_element (PCE) element not present.
- CMAF Fragment durations
    - 1.984 seconds
- Fragments containing one or multiple moof/mdat pairs
    - Fragment is 1 chunk.
    - Fragment contains multiple chunks.

10.3.3 Test Content

The test content for AAC audio is documented in Table 3.

Table 3: Test content for AAC audio

ID	PN #	CMAF brand	Codec	brand	trun	elst	drc	PCE	Bitrate kbits/s	Sample Rate kHz	Channel	CMAF Fragment durations (s)	Chunks	Encryption	Duration (s)
at1	PN01	caac	mp4a.40.2	cmfc, cmf2	v0	Present	absent	absent	128	48	Stereo	1.984	1	unencrypted	30
at2	PN01	caac	mp4a.40.2	cmfc, cmf2	v0	Present	absent	absent	128	48	Stereo	1.984	4	unencrypted	30
at3	PN01	caac	mp4a.40.2	cmfc, cmf2	v1	Present	absent	absent	128	48	Stereo	1.984	1	unencrypted	30
at4	PN01	caac	mp4a.40.2	cmfc, cmf2	v1	Absent	absent	absent	128	48	Stereo	1.984	1	unencrypted	30
at5	PN01	caac	mp4a.40.2	cmfc, cmf2	v0	Present	absent	absent	128	48	Stereo	1.984	1	Clearkey	30
at6	PN01	caac	mp4a.40.5	cmfc, cmf2	v0	Present	absent	absent	128	48	Stereo	1.984	1	unencrypted	30
at7	PN01	caac	mp4a.40.29	cmfc, cmf2	v0	Present	absent	absent	128	48	Stereo	1.984	1	unencrypted	30
at8	PN01	caaa	mp4a.40.2	cmfc, cmf2	v0	Present	absent	absent	128	48	Stereo	1.984	1	unencrypted	30
at9	PN01	caaa	mp4a.40.5	cmfc, cmf2	v0	Present	absent	absent	128	48	Stereo	1.984	1	unencrypted	30
at10	PN01	caaa	mp4a.40.29	cmfc, cmf2	v0	Present	absent	absent	128	48	Stereo	1.984	1	unencrypted	30
at11	PN01	camc	mp4a.40.2	cmfc, cmf2	v0	Present	absent	absent	128	48	5.1	1.984	1	unencrypted	30
at12	PN01	camc	mp4a.40.5	cmfc, cmf2	v0	Present	absent	absent	128	48	5.1	1.984	1	unencrypted	30
at13	PN03	caac	mp4a.40.2	cmfc, cmf2	v0	Present	absent	absent	128	48	Stereo	1.984	1	unencrypted	10

at 14	PN 04	caac	mp4a.40.2	cmfc, cmf2	v0	Present	absent	absent	128	48	Stereo	1.984	1	unencrypted	5.76

NOTE: The table is a dump of the online version of the AAC test matrix at the time of the technical freeze of this version of the specification. The online version will be updated and maintained with bug fixes and additional content.

10.3.4 Playback Requirements

General CMAF audio track playback includes:

- The requirement to support following playback requirements as documented in clause 8:
  - 8.2 Sequential Track Playback for all prioritized content options
  - 8.3 Random Access to Fragment for all prioritized content options
  - 8.4 Random Access to Time for all prioritized content options
  - 8.6 Regular Playback of Chunked Content for all prioritized content options
  - 8.7 Regular Playback of Chunked Content, non-aligned append for all prioritized content options
  - 8.13 Restricted Splicing of Encrypted Content for all prioritized content options
  - 8.14 Sequential Playback of Encrypted and Non-Encrypted Baseline Content for all prioritized content options
  - 8.15 Source Buffer Re-Initialization (without changeType) for all prioritized content options
  - 8.16 Source Buffer Re-Initialization (with changeType) for all prioritized content options
  - 8.17 Buffer Underrun and Recovery for all prioritized content options
  - 8.18 Truncated Playback and Restart for all prioritized content options
  - 8.19 Low-Latency (1): Initialization for all prioritized content options
  - 8.20 Low-Latency"   Playback over Gaps for all prioritized content options
  - 8.21 MSE – AppendWindow for all prioritized content options
  - 8.22 Low-Latency (2): Short Buffer Playback for all prioritized content options
  - 8.23 Random access from one place in a stream to a different place in the same stream for all prioritized content options
- The recommendation to support the following playback requirements as documented in clause 8:
  - 8.8 Playback over WAVE Baseline Splice Constraints
  - 8.9 Out-Of-Order Loading for all prioritized content options
  - 8.10 Overlapping Fragments for all prioritized content options
- The option to support all applicable requirements from clause 8 for non-prioritized content options.

10.3.5 Test Cases

The test cases for AAC audio are documented in Table 4 where the content options are defined in section 10.3.3.

Table 4: Test cases for AAC audio

Test Code / Audio Content ID	at1	at2	at3	at4	at5	at6	at7	at8	at9	at10	at11	at12	at13	at14
8.2 Sequential Track Playback	X		X	X		X	X	X	X	X	X	X		
8.3 Random Access to Fragment	X													

Test Code / Audio Content ID	at1	at2	at3	at4	at5	at6	at7	at8	at9	at10	at11	at12	at13	at14
8.4 Random Access to Time	X													
8.6 Regular Playback of Chunked Content		X												
8.7 Regular Playback of Chunked Content, non-aligned append		X												
8.8 Playback over WAVE Baseline Splice Constraints													X	X
8.9 Out-Of-Order Loading	X													
8.10 Overlapping Fragments	X													
8.12 Playback of Encrypted Content					X									

NOTE: The attached file is a dump of the online version of the AAC test cases at the time of the technical freeze of this version of the specification. The online version will be updated and maintained with bug fixes and additional content.

10.4 Subtitle

10.4.1 General

For subtitles, general playback requirements are tested with the media profile CMAF IMSC Profile as defined in [WAVE-CON], clause 4.4 and [CMAF].

For instructions on subtitle testing, refer to clause 13.

10.5 Encrypted Content

10.5.1 General

For encrypted content, general playback requirements are tested with the media profile CMAF AVC HD as defined in [WAVE-CON], clause 4.2 and [CMAF].

The encryption parameters of a capability discovery are driven by both track packaging descriptions and application deployment choices. All media profiles defined in [WAVE-CON] may utilize encryption as outlined by [CMAF] clause 8 with encryption mode considerations described by [WAVE-CON] Annex C.

Application deployment choices will drive the selection of key systems and media key session parameters, further details are provided as part of parameter setting descriptions.

Capability Discovery and Source Buffer Initialization for this media profile are addressed in clauses 11.2.2 and 11.2.3, respectively.

10.5.2 Capability Discovery

A device supporting encrypted media playback shall support at least one of the following capability discovery mechanisms:

1) The navigator.requestMediaKeySystemAccess() method is called with a sequence of potential MediaKeySystemConfiguration objects for a keySystem and returns a MediaKeySystemAccess object, based on the following parameters for the call:

    a. The keySystem argument is set to the desired key system, an overview of key system identification is detailed in Table 5. The information can be accessed from the **ContentProtection** element in a DASH MPD and/or from the 'pssh' box based on the UUID as specified here: https://dashif.org/identifiers/content_protection/.

    b. The sequence of MediaKeySystemConfiguration objects are based on the following parameters with application choice points providing variations:

        i. label is set to a unique identifier the application can use to relate the chosen configuration back to the supplied options.

        ii. initDataTypes is set to an array of values based on the target key system, an overview of key system initialization data type values is detailed in Table 6.

        iii. sessionTypes is set to "temporary" as the application does not require the persistence of encryption key data by the CDM for this media session.

        iv. distinctiveIdentifier is set to "optional".

        v. persistentState is set to "optional".

        vi. audioCapabilities is set to a sequence of MediaKeySystemMediaCapability objects based on the following parameters with target audio media profiles points providing variations:

            1. contentType is set to the MIME type containing the codec argument for a target audio media profile as defined in the MPD.

            2. robustness is set to the CDM specific robustness description desired by the application for audio tracks. In the testing case, the information is blank unless robustness signaling is provided in the manifest/MPD.

        vii. videoCapabilities is set to a sequence of MediaKeySystemMediaCapability objects based on the following parameters with target video media profiles providing variations:

            1. contentType is set to the MIME type containing the codec argument for a target video media profile as defined in the MPD.

            2. robustness is set to the CDM specific robustness description desired by the application for video tracks. In the testing case, the information is blank unless robustness signaling is provided in the manifest/MPD.

2) The mediaCapabilities.decodingInfo() method is called with the keySystemConfiguration attribute present and returns a MediaCapabilitiesInfo object with the Boolean 'supported' attribute as TRUE, based on the following parameters for the keySystemConfiguration:

    a. keySystem is set to the desired key system, an overview of key system identification is detailed in Table 5.

    b. initDataType is set based on the targeted key system, an overview of key system initialization data type values is detailed in Table 6.

    c. sessionTypes is set to an array containing "temporary" as the application does not require the persistence of encryption key data by the CDM for this media session.

    d. distinctiveIdentifier is set to "optional".

e. `persistentState` is set to `"optional"`.

f. `audio` is set to a `KeySystemTrackConfiguration` where:

   i. `robustness` is set to the CDM specific robustness description desired by the application for audio tracks.

   ii. `encryptionScheme` is set as follows:

      1. if the CMAF Header contains a `schm`-box

         a. `"cenc"` if the `scheme_type` is `"cenc"`

         b. `"cbcs"` if the `scheme_type` is `"cbcs"`

      2. else if a DASH MPD is available, then the value is set to the `@value` of the **`ContentProtection`** element with `@schemeIdUri="urn:mpeg:dash:mp4protection:2011"` associated with audio tracks.

      3. else if an HLS M3U8 is available with `METHOD=SAMPLE-AES` for a declared `EXT-X-KEY`, then the value is set to `"cbcs"`.

g. `video` is set to a `KeySystemTrackConfiguration` where:

   i. `robustness` is set to the CDM-specific robustness description desired by the application for video tracks.

   ii. `encryptionScheme` is set as follows:

      1. if the CMAF Header contains a `schm`-box

         a. `"cenc"` if the `scheme_type` is `"cenc"`.

         b. `"cbcs-1-9"` if the `scheme_type` is `"cbcs"` and the encrypt:skip pattern is 1:9.

         c. `"cbcs"` if the `scheme_type` is `"cbcs"` and the `encrypt:skip` pattern is not 1:9 or is otherwise unknown.

      2. else if a DASH MPD is available, then the value is set to the `@value` of the `ContentProtection` element with `@schemeIdUri="urn:mpeg:dash:mp4protection:2011"` associated with audio tracks.

      3. else if an HLS M3U8 is available with `METHOD=SAMPLE-AES` for a declared `EXT-X-KEY`, then the value is set to `"cbcs-1-9"`.

Table 5: keySystem values and functional variants for well-known DRM systems[1]

DRM System	keySystem Value	Description
Apple FairPlay	`com.apple.fps.1_0`	FairPlay key system for playback of encrypted content in Type 1 player mode (i.e. `src=m3u8`).

[1] https://dashif.org/identifiers/content_protection/

DRM System	keySystem Value	Description
	`com.apple.fps.2_0`	FairPlay key system for playback of encrypted content in Type 3 player mode (i.e., MSE/EME).
Google Widevine	`com.widevine.alpha`	Standard Widevine key system.
Microsoft PlayReady	`com.microsoft.playready`	Original and most widely available PlayReady implementation via EME, but is not fully specification compliant.
	`com.microsoft.playready.hardware`	Same as above, but explicitly requests hardware backed implementations of PlayReady only.
	`com.microsoft.playready.recommendation`	New and fully EME compliant implementation of PlayReady, strongly recommended for usage where available by Microsoft.
W3C Clear Key	`org.w3.clearkey`	W3C defined common key system utilizing in the clear encryption keys [W3C EME], clause 9.1.

Table 6: Initialization data types and their well-known DRM System compatibility

Initialization Data Type	Compatible DRM Systems
`cenc`	• Microsoft PlayReady • Google Widevine • W3C Clear Key
`sinf`	• Apple FairPlay

10.5.3 Content Options

For encrypted content the following source content options are identified.

Source Content

- Audio
- Video

Encrypted Content

- *Scheme:*
  - *cbcs as defined in Content specification, "Web Application Video Ecosystem - Content Specification (CTA-5001-D), clause 4.5:*
    - *for video using cbcs with partial encryption.*
    - *For other tracks full encryption.*
  - *cenc*
- 8.2.1 The TrackEncryptionBox
  - *version 1*
  - version zero
- 8.2.2.1 SampleEncryptionBox
  - *Present*
  - Absent

- 8.2.2.1 Sample Auxiliary information
    - Absent
    - *Present*
- *8.2.2.1* SampleAuxiliaryInformationSizesBox
    - Absent
    - *Present*
- 8.2.2.1 Per_Sample_IV_Size is
    - Zero
    - Non-zero
- 8.2.2.3 pssh
    - *Absent from CMAF Header and CMAF Fragments.*
    - Version 0 present in CMAF Header.
    - Version 1 present in CMAF Header.
    - Version 1 present in CMAF Fragments.
    - Presence on multiple levels.
    - Presence of multiple pssh boxes.
- 8.2.3.1 Any additional keys described by sample groups may be stored in version 1 `ProtectionSystemSpecificHeaderBoxes` in CMAF fragments, identifying the contained KID(s), protected by DRM specific methods.
    - *No keys stored.*
    - Additional keys stored.
- 8.2.3.2 Clear Samples
    - *Sample groups not indicating unprotected media samples, as specified in ISO/IEC 23001-7.*
    - Sample groups indicating unprotected media samples, as specified in ISO/IEC 23001-7.

DRM System

- https://github.com/Dash-Industry-Forum/ClearKey-Content-Protection
- Use TLS
- Test Manifest
    - Add bullet 1 from github.
    - Also add mp4protection scheme and the appropriate encryption ‘cbcs’/’cenc’.
- Test Runner implementation would follow what is in 2 and 3.
- Can we use the keyID for the content key?
    - Option 1:
        - Provide the key in a separate file that is linked from the manifest according to 1.
        - Format of file conforms to json object as defined in step 3.
    - Option 2:
        - We add the key directly to the manifest.
- Open Source
    - gpac or shaka packager

Media Profile specific tests for encrypted content are document below.

10.5.4 Test Content

The test cases for AVC encrypted video are included in the attached sheet as 'WAVE test content _ test code sparse Matrix - AVC.xlsx'.

> NOTE: The attached file is a dump of the online version of the AVC test matrix at the time of the technical freeze of this version of the specification. The online version will be updated and maintained with bug fixes and additional content.

10.5.5 Playback Requirements

Support for encrypted CMAF video track playback includes:

- The requirement to support the following playback requirements as documented in clause 8:
    - 8.12 Playback of Encrypted Content for all prioritized encrypted content options
    - 8.13 Restricted Splicing of Encrypted Content
    - 8.14 Sequential Playback of Encrypted and Non-Encrypted Baseline Content

10.5.6 Test Cases

The test cases for encrypted AVC video are included in the attached sheet as 'WAVE test content _ test code sparse Matrix - AVC.xlsx'.

> NOTE: The attached file is a dump of the online version of the AVC test matrix at the time of the technical freeze of this version of the specification. The online version will be updated and maintained with bug fixes and additional content.

10.6 Presentation Playback

10.6.1 General

For presentation playback as defined in clause 9, the playback of CTA WAVE programs and CMAF Presentations is tested. For this purpose, it is assumed that two CMAF Switching Sets of two different media types are available and they are played jointly and synchronized according to the requirements.

10.6.2 Content Options

For presentation playback as defined in clause 9, the playback of CTA WAVE programs and CMAF Presentations is tested. For this purpose, it is assumed that two CMAF Switching Sets of two different media types are available and they are played jointly and synchronized according to the requirements.

Generally, many combinations of the content options documented in clause 10.2.3 for video and 10.3.3 for audio may be considered. However, a few prioritized test combinations are as follows:

- Unaligned segment durations between Audio and video – e.g., video is 2 seconds and audio is 1.984 seconds.

- Edit lists are present in audio, not in video.
- Edit lists are present in audio and video, but the duration in the edit list is different.
- Video is encrypted and audio is non-encrypted.

10.6.3 Test Content

As test content, combinations for existing AVC and AAC test content are used.

10.6.4 Playback Requirements

Support for general CMAF presentation playback includes:

- The requirement to support following playback requirements as documented in clause 9:
  - 9.2 Regular Playback of a CMAF Presentation for prioritized content options
  - 9.3 Random Access of a WAVE Presentation for prioritized content options
  - 9.6 Long Duration Playback for prioritized content options
- The recommendation to support following playback requirements as documented in clause 9:
  - 9.2 Regular Playback of a CMAF Presentation for any content options
  - 9.3 Random Access of a WAVE Presentation for any content options
  - 9.6 Long Duration Playback for any content options

10.6.5 Test Cases

Table 7 provides the test cases for WAVE presentation playback, where the AVC video content options are defined in section 10.2.4, and the AAC audio content options are defined in section 10.3.3.

Table 7: Test cases for WAVE Presentation Playback

Test Code	Video Content	Audio Content
9.2 Regular Playback of a CMAF Presentation	cfhd_sets/12.5_25_50/t1	caac_sets/aac_lc/at1
9.2 Regular Playback of a CMAF Presentation	cfhd_sets/14.985_29.97_59.94/t1	caac_sets/aac_lc/at1
9.2 Regular Playback of a CMAF Presentation	cfhd_sets/15_30_60/t1	caac_sets/aac_lc/at1
9.3 Random Access of a WAVE Presentation	cfhd_sets/12.5_25_50/t1	caac_sets/aac_lc/at1
9.3 Random Access of a WAVE Presentation	cfhd_sets/14.985_29.97_59.94/t1	caac_sets/aac_lc/at1
9.3 Random Access of a WAVE Presentation	cfhd_sets/15_30_60/t1	caac_sets/aac_lc/at1

10.7 Presentation Splicing

10.7.1 General

For presentation splicing, clause 9, the sequential playback of CTA WAVE programs and CMAF Presentations is tested. For this purpose, it is assumed that four CMAF Switching Sets are available, two for media type audio and two for media type video. Two different media types are played jointly up to a certain time and after some time, the other two media types are played jointly.

10.7.2 Content Options

Generally, many combinations of the content options documented in clause 10.2.3 for video and 10.3.3 for audio may be considered. However, a few prioritized test combinations are as follows:

- Same codec in both CTA WAVE presentations.
- Different codecs in either CTA WAVE presentations.
- Encrypted CTA WAVE presentation followed by unencrypted.
- Unencrypted CTA WAVE presentation followed by encrypted.

10.7.3 Test Content

As test content, combinations for existing AVC and AAC test content are used.

10.7.4 Playback Requirements

Support for Presentation Splicing:

- Shall support following playback requirements as documented in clause 9:
  - 9.4 Splicing of WAVE Program with Baseline Constraints for prioritized content options
- Should support following playback requirements as documented in clause 9:
  - 9.4 Splicing of WAVE Program with Baseline Constraints for any content options

10.7.5 Test Cases

Table 8 provides the test cases for WAVE presentation splicing playback, where the AVC video content options are defined in section 10.2.4, and the AAC audio content options are defined in section 10.3.3.

Table 8: Test cases for WAVE Presentation Splicing Playback

Test Code	Video Content	Audio Content
9.4 Splicing of WAVE Program with Baseline Constraints	cfhd_sets/12.5_25_50/splice_main cfhd_sets/12.5_25_50/splice_ad	caac_sets/aac_lc/at13 caac_sets/aac_lc/at14

11 VIDEO MEDIA PROFILES

11.1 General

11.1.1 Introduction

The video media profiles are defined in [WAVE-CON], clause 4.2.

11.1.2 Capability Discovery Options

A device supporting a media profile shall support at least one of the following capability discovery mechanisms:

1) The "is supported type query" for the media profile with argument `video/mp4 profiles="<brand>"` results in a `yes` with `<brand>` specific for each profile. An overview is provided in Table 9.

2) The "is supported type query" for the `codecs` with an argument `video/mp4 codecs="<codecs>"` results in a yes as defined in [WAVE-CON], Table 1. The value is obtained from the according to the sample entry `codingname` field of the CMAF Principal Header.

3) The playback can be started by a CMAF header that conforms to the media profile under consideration.

4) The `mediaCapabilities.decodingInfo()` media is called with a video configuration object and returns the Boolean `'supported'` attribute is `TRUE` for the `MediaCapabilitiesInfo`, based on the following parameters settings:

   a. The `contentType` with argument `video/mp4 codecs="<codecs>"` as defined in [WAVE-CON], Table 1. The value is obtained from the according to the sample entry `codingname` field of the CMAF Principal Header.

   b. `width` is set to the `width` in the CMAF `TrackHeaderBox` of the CMAF Principal Header of the CMAF Switching Set.

   c. `height` is set to the `height` in the CMAF `TrackHeaderBox` of the CMAF Principal Header of the CMAF Switching Set.

   d. `bitrate` is set as follows:

      i. If the CMAF Header contains `btrt`-box, and the box contains the `MaxBitrate` value, then the bitrate is set to this value.

      ii. Else if a DASH MPD is available, then the value is set to the value of `@bandwidth`,

      iii. Else if an HLS M3U8 is available, then the value is set to the value of the `BANDWIDTH` value,

      iv. Else the value is set to `<max-bitrate>` (corresponding to the highest bitrate of this profile level combination) with `<max-bitrate>` specific for each profile. An overview is provided in Table 9.

   e. `framerate` is set as follows:

      i. If the following information is available:
         1. Get `moov/mvex/trex/default_sample_duration` (e.g., 1001) from CMAF Principal Header.
         2. Get `moov/trak/mdia/mdhd/timescale` (e.g., 24000) from CMAF Principal Header.
         3. The value is set as `"timescale/default_sample_duration"` (e.g., 24000/1001).

      ii. Else if a DASH MPD is available, then the value is set to the computed value of the `@framerate` attribute.

      iii. Else if an HLS M3U8 is available, then the value is set to the value of the `FRAME-RATE` value.

      iv. Else some media profile specific settings may be documents, for example as part of the Sequence Parameter Sets (SPS).

      v. Else the value is set to `<max-framerate>` specific for each profile. An overview is provided in Table 9.

   f. `hasAlphaChannel` is not present.

   g. `hdrMetadataType` is set according to the media profile. For an overview see Table 9.

   h. `colorGamut` is set according to the media profile. For an overview see Table 9.

   i. `transferFunction` is set according to the media profile. For an overview see Table 9.

Table 9: Overview of Capability codes for different video media profiles

Media Profile Name	<brand>	<codecs>	<max-framerate>	hdrMetadataType	colorGamut	transferFunction
HD	`cfhd`	`avc1.640028` `avc3.640028`	60	not present	`sRGB`	`sRGB`
HHD10	`chh1`	`hev1.2.4.L123.B0` `hvc1.2.4.L123.B0`	60	not present	`sRGB`	`sRGB`
UHD10	`cud1`	`hev1.2.4.L153.B0` `hvc1.2.4.L153.B0`	60	not present	`sRGB` or `rec2020`	`sRGB`
HDR10	`chd1`	`hev1.2.4.L153.B0` `hvc1.2.4.L153.B0`	60	`smpteSt2086` or may be absent	`rec2020`	`pq`
HLG10	`clg1`	`hev1.2.4.L153.B0` `hvc1.2.4.L153.B0`	60	may be absent	`rec2020`	`pq`

11.2 Media Profile: CMAF AVC HD (`'cfhd'`)

11.2.1 Introduction

The media profile CMAF AVC HD is defined in [WAVE-CON], clause 4.2 and [CMAF].

11.2.2 Capability Discovery Options

The generic capability discovery from clause 11.1.2 applies with the following specific aspects:

- `<brand>` is set to `"cfhd"`.
- `<codecs>` is set to `"avc1.640028"` or `"avc3.640028"`.
- `<max-bitrate>` is set to 25,000,000.
- `<max-framerate>` is set to 60.
- `hdrMetadataType` is not present.
- `colorGamut` is set to `srgb`.
- `transferFunction` is set to `srgb`.

11.2.3 Source Buffer Initialization Requirements

If no other video source buffer is available, then the device shall support the creation of a source buffer with any of the codecs parameters `video/mp4 codec="avc1.640028"` or `video/mp4 codec="avc3.640028"` results in a `yes` as defined in [WAVE-CON], Table 1 for this profile.

11.2.4 Content Options

See clause 10.2.2.

11.2.5 Test Content

See clause 10.2.3.

11.2.6 Playback Requirements

See clause 10.2.4.

11.2.7 Test Cases

See clause 10.2.5.

11.3 Media Profile: CMAF HEVC HHD10 ('chh1')

11.3.1 Introduction

The media profile CMAF HEVC HHD10 ('chh1') is defined in [WAVE-CON], clause 4.2 and [CMAF].

11.3.2 Capability Discovery Options

The generic capability discovery from clause 11.1.2 applies with the following specific aspects:

- `<brand>` is set to `"chh1"`
- `<codecs>` is set to `"hev1.2.4.L123.B0"` or `"hvc1.2.4.L123.B0"`
- `<max-bitrate>` is set to 20,000,000.
- `<max-framerate>` is set to 60.
- `hdrMetadataType` is not present.
- `colorGamut` is set to `srgb`.
- `transferFunction` is set to `srgb`.

11.3.3 Source Buffer Initialization Requirements

If no other video source buffer is available, then the device shall support the creation of a source buffer with any of the codecs parameters `video/mp4 codec="hev1.2.4.L123.B0"` or `video/mp4 codec="hvc1.2.4.L123.B0"` results in a `yes` as defined in [WAVE-CON], Table 1 for this profile.

11.3.4 Content Options

The content options are based on clause 10.2.2, but only a subset is recommended to be tested as general constraints are expected to be covered by the tests in clause 10.2.

Proposed CMAF Options (see clause 7 of CMAF [CMAF])

- Compositions offsets and Timing:
  - `'cmfc'`
    - 7.5.17 Track Run Box (`'trun'`)
      - v1 – combined with SAP type 1
    - 7.5.13 Edit List Box (`'elst'`)

            - Absent
        - ***Two options for default flags***
            - A: `default_sample_flags`, `sample_flags` and `first_sample_flags` neither set in the `TrackFragmentHeaderBox` nor `TrackRunBox`
            - B: `default_sample_flags`, `sample_flags` and `first_sample_flags` set in the `TrackFragmentHeaderBox` and `TrackRunBox`

- 7.4.5 Event Message Box (`'emsg'`)
    - *Absent*

Source Content Options

- 50Hz video
- 60Hz video, 60/1.001 Hz
- ***Priority is given to 1920x1080p50 and 1920x1080p60/1.001.***
- 16:9 picture aspect ratio square pixel [CMAF], clause 9.2.3 and 9.4.2.2.2.
- At least one source sequence with another picture aspect ratio, non-square pixel with conformance cropping following the example in Annex C.8 of [CMAF] specification.

Encoding and Packaging options

- SEI and VUI
    - ***with (W) and without (W/O) picture timing SEI message***
    - ***with (W) and without (W/O) with and without VUI timing information***
- *Sample entry, see CMAF clause 9.4.1.2.* ***Two options***
    - 'hvc1' sample entry type (parameter sets within the CMAF Header)
    - 'hev1' sample entry type (parameter sets within the movie fragment header)
- Initialization Constraints.
    - Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4.
- CMAF Fragment durations
    - *2 seconds*
- Fragments containing **one or multiple moof/mdat pairs –** ***two options***
    - *A: Fragment is 1 chunk*
    - *B: Fragment contains multiple chunks (p-frame to p-frame with b-frames)*
- **Switching Set encoding following CMAF, Annex C recommendation. The following options exist:**
    - *No subsampling.*
    - *Spatial subsampling of video according to CMAF, Annex C.2 recommendation.*
    - *Spatial and temporal subsampling and scaling of video according to CMAF, Annex C.1 recommendation for each 50 Hz and 60 Hz family.*
    - Regular encoding or CMAF, Annex C.9.4 recommendation for low-latency live.

- B.2.4
    - No SEI messages present.
- B.3.3.2
    - Single VPS.
- B.3.3.3.2
    - colour_description_flag set to 1.
    - vui_time_scale and vui_num_units_in_tick set to constant.
- B.5
    - Settings according to the requirements of the profile 'chh1' in Table B.1 of ISO/IEC 23000-19.

11.3.5 Test Content

For initial test content definition, please check here https://github.com/cta-wave/Test-Content-Generation/blob/master/Instructions/chh1.md (Issue) and Table 10.

Table 10: Test content overview for `'chhd'`

Number	Default Flags	Sample entry	SEI Timing	VUI Timing	Chunks	Subsampling	Source content
chh1_1	set	hvc1	w/o	w/o	1	no	*1920x1080p50*
chh1_2	not set	hev1	w/o	w/o	1	no	*1920x1080p50*
chh1_3	set	hvc1	w/o	w/o	10	no	*1920x1080p50*
chh1_4	set	hvc1	w/o	w/o	1	2 spatial (1080/720) 2 temporal (50/25)	*1920x1080p50*
chh1_5	set	hvc1	w/o	w/o	1	2 spatial (1080/720) 2 temporal (30/60)	*1920x1080p @ 60/1.001*
chh1_6			w	w			

11.3.6 Playback Requirements

This is for further study.

11.3.7 Test Cases

This is for further study.

11.4 Media Profile: CMAF HEVC UHD10 ('cud1')

11.4.1 Introduction

The media profile CMAF HEVC UHD10 is defined in [WAVE-CON], clause 4.2 and [CMAF].

11.4.2 Capability Discovery Options

The generic capability discovery from clause 11.1.2 applies with the following specific aspects:

- `<brand>` is set to "`cud1`".
- `<codecs>` is set to "`hev1.2.4.L153.B0`" or "`hvc1.2.4.L153.B0`".
- `<max-bitrate>` is set to 40,000,000.
- `<max-framerate>` is set to 60.
- `hdrMetadataType` is not present.
- `colorGamut` is set as follows:
    a. if the Sequence Parameter Set (SPS) is present in the CMAF Principal Header in the `HEVCDecodingConfigurationRecord()`, and the `vui_parameters_present_flag` is set to 1 (as required by CMAF, see ISO/IEC 23000-19, clause B.3.3.4), then if
        i. `colour_primaries` is set to 1, the value is set to `'srgb'`.
        ii. `colour_primaries` is set to 9, the value is set to `'rec2020'`.
    b. else if a DASH MPD is available, and the CMAF Track has assigned a descriptor with `@schemeIdURI` set to `urn:mpeg:mpegB:cicp:ColourPrimaries` and then if the `@value` of this descriptor:
        i. is set to 1, the value us set to `'srgb'`.
        ii. is set to 9, the value us set to `'rec2020'`.
    c. else the value is set to `'rec2020'`
- `transferFunction` is set to `srgb`.

11.4.3 Source Buffer Initialization Requirements

If no other video source buffer is available, then the device shall support the creation of a source buffer with any of the codecs parameters `video/mp4 codec="hev1.2.4.L153.B0"` or `video/mp4 codec="hvc1.2.4.L153.B0"` results in a `yes` as defined in [WAVE-CON], Table 1 for this profile.

11.4.4 Content Options

The content options are based on clause 10.2.2, but only a subset is recommended to be tested as general constraints are expected to be covered by the tests in clause 10.2.

Proposed CMAF Options (see clause 7 of CMAF [CMAF])

- Compositions offsets and Timing:
    - `'cmfc'`
        - 7.5.17 Track Run Box (`'trun'`)
            - v1 – combined with SAP type 1 .
        - 7.5.13 Edit List Box (`'elst'`)
            - Absent

        - Two options for default flags:
            - `default_sample_flags` set in `TrackFragmentHeaderBox` and `first_sample_flags` set in `TrackRunBox`. With `sample_flags` not used.
            - `sample_flags` set in `TrackRunBox` for every sample. With `default_sample_flags` and `first_sample_flags` not used.
- 7.4.5 Event Message Box (`'emsg'`)
    - *Absent*

Source Content Options

- 50Hz video
- 60Hz video, 60/1.001 Hz
- *Priority is given to 3840x2160p50 and 3840x2160p60*
- *BT.709 and BT.2020*
- 16:9 picture aspect ratio square pixel [CMAF], clause 9.2.3 and 9.4.2.2.2.
- At least one source sequence with another picture aspect ratio, non-square pixel with conformance cropping following the example in Annex C.8 of [CMAF] specification.

Encoding and Packaging options

- SEI and VUI
    - *with and without picture timing SEI message.*
    - *with and without VUI timing information.*
- Sample entry, see CMAF clause 9.4.1.2. Two options:
    - *'hvc1' sample entry type (parameter sets within the CMAF Header) .*
    - *'hev1' sample entry type (parameter sets within the movie fragment header).*
- Initialization Constraints
    - Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4.
- CMAF Fragment durations
    - *2 seconds.*
- Fragments containing one or multiple moof/mdat pairs – two options:
    - *Fragment is 1 chunk.*
    - *Fragment contains multiple chunks (p-frame to p-frame with b-frames).*
- Switching Set encoding following CMAF, Annex C.9 recommendation. At least one of each of the following options:
    - Spatial subsampling of video according to CMAF, Annex C.2 recommendation.
    - Spatial and temporal subsampling and scaling of video according to CMAF, Annex C.1 recommendation for each 50 Hz and 60 Hz family.
    - CMAF, Annex C.9.4 recommendation for low-latency live.

- B.2.4
    - No SEI messages.
- B.3.3.2
    - Single VPS.
- B.3.3.3.2
    - `colour_description_flag` set to 1.
    - `vui_time_scale` and `vui_num_units_in_tick` set to constant.
- B.5
    - Settings according to the requirements of the profile cuh1 in Table B.1.

11.4.5 Test Content

For initial test content definition, please check here https://github.com/cta-wave/Test-Content-Generation/blob/master/Instructions/cud1.md (Issue) and Table 11.

Table 11: Test content overview for 'cud1'

Number	Default flags	Sample entry	chunks	Spatial and temporal supsampling	Source content
hevc21	yes	hvc1	1	no	*3840x2160p50*
hevc22	no	hev1	1	no	*3840x2160p50*
hevc23	yes	hvc1	10	no	*3840x2160p50*
hevc24	yes	hvc1	1	3 spatial (2160, 1440, 1080) 2 temporal (50 & 25)	*3840x2160p50*
hevc25	yes	hvc1	1	3 spatial (2160, 1440, 1080) 2 temporal (30 & 60)	*3840x2160p @ 60/1.001*

11.4.6 Playback Requirements

This is for further study.

11.4.7 Test Cases

This is for further study.

11.5 Media Profile: CMAF HEVC HDR10 ('chd1')

11.5.1 Introduction

The media profile CMAF HEVC UHD10 is defined in [WAVE-CON], clause 4.2 and [CMAF].

11.5.2 Capability Discovery Options

The generic capability discovery from clause 11.1.2 applies with the following specific aspects:

- `<brand>` is set to `"chd1"`.

- `<codecs>` is set to `"hev1.2.4.L153.B0"` or `"hvc1.2.4.L153.B0"`.
- `<max-bitrate>` is set to 40,000,000.
- `<max-framerate>` is set to 60.
- if the `HEVCDecoderConfigurationRecord()` in the CMAF Principal Header includes an SEI NAL unit for `Principaling_display_colour_volume`, SEI `payloadType=137` then `hdrMetadataType` should be present and set to `'smpteSt2086'`, otherwise the `hdrMetadataType` may be present.
- `colorGamut` is set to `'rec2020'`.
- `transferFunction` is set to `'pq'`.

11.5.3 Source Buffer Initialization Requirements

If no other video source buffer is available, then the device shall support the creation of a source buffer with any of the codecs parameters `video/mp4 codec="hev1.2.4.L153.B0"` or `video/mp4 codec="hvc1.2.4.L153.B0"` results in a `yes` as defined in [WAVE-CON], Table 1 for this profile.

11.5.4 Content Options

The content options are based on clause 10.2.2, but only a subset is recommended to be tested as general constraints are expected to be covered by the tests in clause 10.2.

Proposed CMAF Options (see clause 7 of CMAF [CMAF])

- Compositions offsets and Timing:
  - `'cmfc'`
    - 7.5.17 Track Run Box (`'trun'`)
      - v1 – combined with SAP type 1
    - 7.5.13 Edit List Box (`'elst'`)
      - Absent
    - Two options for default flags:
      - `default_sample_flags` set in `TrackFragmentHeaderBox` and `first_sample_flags` set in `TrackRunBox`. With `sample_flags` not used.
      - `sample_flags` set in `TrackRunBox` for every sample. With `default_sample_flags` and `first_sample_flags` not used.
- 7.4.5 Event Message Box (`'emsg'`)
  - Absent

Source Content Options

- 50Hz video
- 60Hz video, 60/1.001 Hz
- *Priority is given to 3840x2160p50 and 3840x2160p60*
- HDR PQ10 content needs to be used with static metadata present

- 16:9 picture aspect ratio square pixel [CMAF], clause 9.2.3 and 9.4.2.2.2.
- At least one source sequence with another picture aspect ratio, non-square pixel with conformance cropping following the example in Annex C.8 of [CMAF] specification.

Encoding and Packaging options

- SEI and VUI
  - With and without picture timing SEI message.
  - With and without VUI timing information.
- Sample entry, see CMAF clause 9.4.1.2. Two options:
  - 'hvc1' sample entry type (parameter sets within the CMAF Header) .
  - 'hev1' sample entry type (parameter sets within the movie fragment header).
- Initialization Constraints
  - Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4.
- CMAF Fragment durations
  - 2 seconds.
- Fragments containing one or multiple moof/mdat pairs – two options:
  - Fragment is 1 chunk.
  - Fragment contains multiple chunks (p-frame to p-frame with b-frames).
- Switching Set encoding following CMAF, Annex C.9 recommendation. At least one of each of the following options:
  - Spatial subsampling of video according to CMAF, Annex C.2 recommendation.
  - Spatial and temporal subsampling and scaling of video according to CMAF, Annex C.1 recommendation for each 50 Hz and 60 Hz family.
  - CMAF, Annex C.9.4 recommendation for low-latency live.
- B.2.4
  - SEI payloadType 137, mastering_display_colour_volume included.
  - SEI payloadType 144, content_light_level_info included.
- B.3.3.2
  - Single VPS.
- B.3.3.3.2
  - colour_description_flag set to 1.
  - vui_time_scale and vui_num_units_in_tick set to constant.
- B.5
  - Settings according to the requirements of the profile chd1 in Table B.1.

11.5.5 Test Content

For initial test content definition, please check here https://github.com/cta-wave/Test-Content-Generation/blob/master/Instructions/chd1.md (Issue) and Table 12.

Table 12: Test content overview for `'chd1'`

Number	Default flags	Sample entry	chunks	Spatial and temporal supsampling	Source content
hevc21	yes	hvc1	1	no	*3840x2160p50*
hevc22	no	hev1	1	no	*3840x2160p50*
hevc23	yes	hvc1	10	no	*3840x2160p50*
hevc24	yes	hvc1	1	3 spatial (2160, 1440, 1080) 2 temporal (50 & 25)	*3840x2160p50*
hevc25	yes	hvc1	1	3 spatial (2160, 1440, 1080) 2 temporal (30 & 60)	*3840x2160p @ 60/1.001*

11.5.6 Playback Requirements

This is for further study.

11.5.7 Test Cases

This is for further study.

NOTE: The tests for '`chh1`' and '`cud1`' should also be carried out.

11.6 Media Profile: CMAF HEVC HLG10 ('clg1')

11.6.1 Introduction

The media profile CMAF HEVC HLG10 is defined in [WAVE-CON], clause 4.2 and [CMAF].

11.6.2 Capability Discovery Options

The generic capability discovery from clause 11.1.2 applies with the following specific aspects:

- `<brand>` is set to `"clg1"`.
- `<codecs>` is set to `"hev1.2.4.L153.B0"` or `codecs="hvc1.2.4.L153.B0"`.
- `<max-bitrate>` is set to 40,000,000.
- `<max-framerate>` is set to 60.
- `hdrMetadataType` may be absent.
- `colorGamut` is set to `'rec2020'`
- `transferFunction` is set to `'hlg'`.

NOTE: HLG content (i.e., the 'clg1' media profile) always has the HLG transfer function. It may be signaled in one of two ways, depending on whether the creator also wants to use the HLG video with a player only supporting the 'cud1' media profile. For the purposes of querying device capabilities, when asking about the 'clg1' profile, it's only the HLG transfer function that's of interest. (If a player were to find that 'clg1' is *not* supported, then the player could then query 'cud1' and if that's OK then it could still play HLG content that has backwards compatible signaling, albeit not with HDR.

11.6.3 Source Buffer Initialization Requirements

If no other video source buffer is available, then the device shall support the creation of a source buffer with any of the codecs parameters `video/mp4 codec="hev1.2.4.L153.B0"` or `video/mp4 codec="hvc1.2.4.L153.B0"` results in a `yes` as defined in [WAVE-CON], Table 1 for this profile.

11.6.4 Content Options

The content options are based on clause 10.2.2, but only a subset is recommended to be tested as general constraints are expected to be covered by the tests in clause 10.2.

Proposed CMAF Options (see clause 7 of CMAF [CMAF])

- Compositions offsets and Timing
  - `'cmfc'`
    - 7.5.17 Track Run Box (`'trun'`)
      - v1 – combined with SAP type 1
    - 7.5.13 Edit List Box (`'elst'`)
      - Absent
    - Two options for default flags:
      - `default_sample_flags` set in `TrackFragmentHeaderBox` and `first_sample_flags` set in `TrackRunBox`. With `sample_flags` not used.
      - `sample_flags` set in `TrackRunBox` for every sample. With `default_sample_flags` and `first_sample_flags` not used.
- 7.4.5 Event Message Box (`'emsg'`)
  - *Absent*

Source Content Options

- 50Hz video.
- 60Hz video, 60/1.001 Hz.
- *Priority is given to 3840x2160p50 and 3840x2160p60.*
- HDR HLG10 content needs to be used.
- 16:9 picture aspect ratio square pixel [CMAF], clause 9.2.3 and 9.4.2.2.2.
- At least one source sequence with another picture aspect ratio, non-square pixel with conformance cropping following the example in Annex C.8 of [CMAF] specification.

Encoding and Packaging options

- SEI and VUI
  - Without picture timing SEI message.
  - Without VUI timing information.
- Sample entry, see CMAF clause 9.4.1.2. Two options:
  - 'hvc1' sample entry type (parameter sets within the CMAF Header).
  - 'hev1' sample entry type (parameter sets within the movie fragment header).
- Initialization Constraints
  - Regular Switching Set, do not apply CMAF clause 7.3.4.2 and 9.2.11.4.
- CMAF Fragment durations
  - *2 seconds.*
- Fragments containing one or multiple moof/mdat pairs – two options:
  - *Fragment is 1 chunk.*
  - *Fragment contains multiple chunks (p-frame to p-frame with b-frames).*
- Switching Set encoding following CMAF, Annex C.9 recommendation. At least one of each of the following options:
  - Spatial subsampling of video according to CMAF, Annex C.2 recommendation.
  - Spatial and temporal subsampling and scaling of video according to CMAF, Annex C.1 recommendation for each 50 Hz and 60 Hz family.
  - CMAF, Annex C.9.4 recommendation for low-latency live.
- B.2.4
  - If transfer_characteristics = 14, then SEI payloadType 147, alternative_transfer_characteristics.
  - If transfer_characteristics = 18, then no SEI message.
- B.3.3.2
  - Single VPS.
- B.3.3.3.2
  - colour_description_flag set to 1.
  - vui_time_scale and vui_num_units_in_tick set to constant.
- B.5
  - Settings according to the requirements of the profile clg1 in Table B.1.

11.6.5 Test Content

For initial test content definition, please check here https://github.com/cta-wave/Test-Content-Generation/blob/master/Instructions/clg1.md (Issue) and Table 13.

Table 13: Test content overview for `'clg1'`

Number	Default flags	Sample entry	chunks	Spatial and temporal supsampling	transfer_characteristics	Source content
hevc31	yes	hvc1	1	no	14	*3840x2160p50*
hevc32	no	hev1	1	no	14	*3840x2160p50*
hevc33	yes	hvc1	10	no	14	*3840x2160p50*
hevc34	yes	hev1	1	3 spatial (2160, 1440, 1080) 2 temporal (50 & 25)	14	*3840x2160p50*
hevc35	yes	hvc1	1	3 spatial (2160, 1440, 1080) 2 temporal (30 & 60)	14	*3840x2160p @ 60/1.001*
hevc36	yes	hvc1	1	no	18	*3840x2160p50*

11.6.6 Playback Requirements

This is for further study.

11.6.7 Test Cases

This is for further study.

12 AUDIO MEDIA PROFILES

12.1 General

12.1.1 Introduction

The audio media profiles are defined in [WAVE-CON], clause 4.3.

12.1.2 Capability Discovery Options

A device supporting a media profile shall support at least one of the following capability discovery mechanisms:

1) The "is supported type query" for the media profile with argument `video/mp4 profiles="<brand>"` or `audio/mp4 profiles="brand"` results in a `yes` with `<brand>` specific for each profile. An overview is provided in Table 14.
2) The "is supported type query" for the codec with argument `video/mp4 codecs="<codecs>"` or `audio/mp4 codecs="<codecs>"` results in a `yes` with `<codecs>` specific for each profile. An overview is provided in Table 14.
3) The playback can be started by a CMAF header that conforms to the media profile under consideration.
4) The `mediaCapabilities.decodingInfo()` media is called with an audio configuration object and returns the Boolean `'supported'` attribute is `TRUE` for the `MediaCapabilitiesInfo`, based on the following parameters settings:

a. The `contentType` with argument `video/mp4 codecs="<codecs>"` or `audio/mp4 codecs="<codecs>"` results in a `yes` with `<codecs>` specific for each profile. An overview is provided in Table 14.

b. `channels` is set to a channel configuration according to [ISO/IEC CICP] and dependent on the media profile but may be absent.

c. `bitrate` is set as follows:

   i. if the CMAF Header contains `btrt`-box, and the box contains the `MaxBitrate` value, then the bitrate is set to this value.

   ii. else if a DASH MPD is available, then the value is set to the value of `@bandwidth` value,

   iii. else if an HLS M3U8 is available, then the value is set to the value of the `BANDWIDTH` value,

   iv. else the field is not present.

d. `samplerate` is set according to the media profile but may be absent.

e. `spatialRendering` is not determined by any parameter of the profile but is set according to an application logic. If no such information is present, the parameter is absent. If the current output device is an HDMI endpoint, the user agent should only report this configuration as supported if the HDMI endpoint capabilities indicate spatial rendering support. If the current output device are speakers or headphones, the user agent should only report this configuration as supported when virtualization technology is applied.

NOTE: `contentType`, `channels` and `sampleRate` may be either signaled offline (as they are known to the operator) or online, for example in a manifest file [DASH] or in the CMAF header of referenced assets. If the `channels` or `sampleRate` parameters are unknown, the parameters may be omitted.

Table 14: Overview of Capability codes for different audio media profiles (informative)

Media Profile Name	<brand>	<codecs>		
AAC Core	`caac`	mp4a.40.2 mp4a.40.5 mp4a.40.29		
Adaptive AAC Core	`caaa`	mp4a.40.2 mp4a.40.5 mp4a.40.29		
AAC Multichannel	`camc`	mp4a.40.2 mp4a.40.5		
Enhanced AC-3, including AC-3	`ceac`	ec-3 ac-3		
AC-4, Single Stream	`ca4s`	ac-4.02.01.03 ac-4.02.01.02 ac-4.02.01.01 ac-4.02.01.00		
MPEG-H, Single Stream	`cmhs`	mhm1.0x0B mhm1.0x0C mhm1.0x0D		
DTS-HD Audio	`dts1`	dtsc dtse		
USAC Stereo	`casu`	mp4a.40.42		

12.1.3 Source Buffer Initialization Requirements

If no other audio source buffer is available, then the device shall support the creation of a source buffer with any of the codecs parameters `<codecs>` results in a `yes` as defined in [WAVE-CON], Table 2 and `<codecs>` specific for each profile. An overview is provided in Table 14 .

12.1.4 Content Options

No common audio content options are defined.

12.1.5 Test Content

No common audio test content is defined.

12.1.6 Playback Requirements

No common audio playback requirements are defined.

12.1.7 Test Cases

No common audio test cases are defined.

12.2 Media Profile: CMAF AAC Core (`'caac'`)

12.2.1 Introduction

The media profile CMAF Adaptive AAC Core is defined in [WAVE-CON], clause 4.3 and [CMAF].

12.2.2 Capability Discovery Options

The generic capability discovery from clause 12.1.2 applies with the following specific aspects:

- `<brand>` is set to `"caac"`.
- `<codecs>` is set to one of the following: `"mp4a.40.5, mp4a.40.29"` as defined in [WAVE-CON], table 2. The exact value is determined from the sample entry `codingname` field of the CMAF Principal Header or defined by the application logic.
- `channels` is defined by the application logic.
- `sampleRate` is defined by the application logic.

12.2.3 Source Buffer Initialization Requirements

The general Source Buffer Initialization requirements as defined in clause 12.1.3 apply with codecs set to `"mp4a.40.5"` or `"mp4a.40.29"`.

12.2.4 Content Options

The content options as defined in clause 10.3.2 apply.

12.2.5 Test Content

The content options as defined in clause 10.3.3 for `'caac'` apply.

12.2.6 Playback Requirements

The playback requirements as defined in clause 10.3.4 apply.

12.2.7 Test Cases

The test cases as defined in clause 10.3.3 for `'caac'` apply.

12.3 Media Profile: CMAF Adaptive AAC Core ('caaa')

12.3.1 Introduction

The media profile CMAF Adaptive AAC Core is defined in [WAVE-CON], clause 4.3 and [CMAF].

12.3.2 Capability Discovery Options

The generic capability discovery from clause 12.1.2 applies with the following specific aspects:

- `<brand>` is set to `"caaa"`.
- `<codecs>` is set to one of the following: `"mp4a.40.2, mp4a.40.5, mp4a.40.29"` as defined in [WAVE-CON] Table 2. The exact value is determined from the sample entry `codingname` field of the CMAF Principal Header or defined by the application logic.
- `channels` is defined by the application logic.
- `sampleRate` is defined by the application logic.

12.3.3 Source Buffer Initialization Requirements

The general Source Buffer Initialization requirements as defined in clause 12.1.3 apply with codecs set to `"mp4a.40.5"` or `"mp4a.40.29"` .

12.3.4 Content Options

The content options as defined in clause 10.3.2 apply. Specifically, a CMAF Switching Set with two CMAF Tracks of different bitrate ought to be added.

12.3.5 Test Content

The content options as defined in clause 10.3.3 for `'caaa'` apply.

12.3.6 Playback Requirements

The playback requirements as defined in clause 10.3.4 apply.

12.3.7 Test Cases

The test cases as defined in clause 10.3.3 for `'caaa'` apply.

12.4 Media Profile: CMAF AAC Multichannel ('camc')

12.4.1 Introduction

The media profile CMAF AAC Multichannel is defined in [WAVE-CON], clause 4.3 and [CMAF].

12.4.2 Capability Discovery Options

The generic capability discovery from clause 12.1.2 applies with the following specific aspects:

- `<brand>` is set to `"camc"`.
- `<codecs>` is set to one of the following: `"mp4a.40.2, mp4a.40.5"` as defined in [WAVE-CON], Table 2. The exact value is determined from the sample entry `codingname` field of the CMAF Principal Header or defined by the application logic.
- `channels` is defined by the application logic.
- `sampleRate` is defined by the application logic.

12.4.3 Source Buffer Initialization Requirements

The general Source Buffer Initialization requirements as defined in clause 12.1.3 apply with codecs set to `"mp4a.40.5"` or `"mp4a.40.29"`

12.4.4 Content Options

The content options as defined in clause 10.3.2 apply. In particular, 5.1 content needs to be added.

12.4.5 Test Content

The test content as defined in clause 10.3.3 for `'camc'` applies.

12.4.6 Playback Requirements

The playback requirements as defined in clause 10.3.4 apply.

12.4.7 Test Cases

The test cases as defined in clause 10.3.3 for `'camc'` apply.

12.5 Media Profile: Enhanced AC-3, including AC-3 ('ceac')

12.5.1 Introduction

The media profile Enhanced AC-3, including AC-3 is defined in [WAVE-CON], clause 4.3 and [ETSI AC-3].

12.5.2 Capability Discovery Options

The generic capability discovery from clause 12.1.2 applies with the following specific aspects:

- `<brand>` is set to `"ceac"`.
- `<codecs>` is set to one of the following: `"ec-3"`, `"ac-3"` as defined in [WAVE-CON], Table 2.
- `channels` is defined by the application logic.
- `sampleRate` is defined by the application logic.

NOTE: See [DASH] and ETSI TS 102366 v1.4.1, Annex F and Annex I for details on how to extract parameters from manifest files or assets.

12.5.3 Source Buffer Initialization Requirements

The general Source Buffer Initialization requirements as defined in clause 12.1.3 apply with codecs set to `"ec-3"` or `"ac-3"`.

12.5.4 Content Options

Encoding and Packaging options

- Codec: Dolby E-AC-3
- Brands: cmf2, ceac
- CMAF Fragment durations
    - 2.016 seconds
- CMAF Switching Set with 2 Tracks of different bitrates

Source Content Options

- Stereo only, 48 kHz

12.5.5 Test Content

The test content for Dolby E-AC-3 audio is documented in Table 15.

Table 15: Test Content for Enhanced AC-3, including AC-3 ('ceac')

ID	PN#	CMAF brand	Codec	brand	Bitrate kbits/s	Sample Rate kHz	Channel	CMAF Fragment durations (s)	Chunks	Duration (s)
at1	PN01	ceac	ec-3	cmfc, cmf2	128	48	Stereo	2.016	1	30
at2	PN01	ceac	ec-3	cmfc, cmf2	128	48	Stereo	2.016	4	30
at3	PN03	ceac	ec-3	cmfc, cmf2	128	48	Stereo	2.016	1	10
at4	PN04	ceac	ec-3	cmfc, cmf2	128	48	Stereo	2.016	1	5.76

12.5.6 Playback Requirements

The playback requirements for AC-3 are as follows:

- Shall support following playback requirements as documented in clause 8:
    - 8.2 Sequential Track Playback for all prioritized content options
    - 8.3 Random Access to Fragment for all prioritized content options
    - 8.4 Random Access to Time for all prioritized content options
    - 8.5 Switching Set Playback
    - 8.6 Regular Playback of Chunked Content for all prioritized content options
    - 8.7 Regular Playback of Chunked Content, non-aligned append for all prioritized content options
- Should support the following playback requirements as documented in clause 8:
    - 8.8 Playback over WAVE Baseline Splice Constraints
    - 8.9 Out-Of-Order Loading for all prioritized content options
    - 8.10 Overlapping Fragments for all prioritized content options
    - 8.12 Playback of Encrypted Content for all prioritized encrypted content options
    - 8.13 Restricted Splicing of Encrypted Content
    - 8.14 Sequential Playback of Encrypted and Non-Encrypted Baseline Content

12.5.7 Test Cases

The test cases for Dolby E-AC-3 audio are documented in Table 16, where the content options are defined in section 12.5.5.

Table 16: Test Cases for Enhanced AC-3, including AC-3 ('ceac')

Test Code	Content ID
8.2 Sequential Track Playback	at1
8.3 Random Access to Fragment	at1
8.4 Random Access to Time	at1
8.6 Regular Playback of Chunked Content	at2
8.7 Regular Playback of Chunked Content, non-aligned append	at2
8.8 Playback over WAVE Baseline Splice Constraints	at3, at4
8.9 Out-Of-Order Loading	at1
8.10 Overlapping Fragments	at1

If Dolby E-AC-3 audio is tested in a presentation playback, the test cases are documented in Table 17.

Table 17: Test Cases for presentation playback for Enhanced AC-3, including AC-3 ('ceac')

Test Code	Video Content ID	Audio Content ID
9.2 Regular Playback of a CMAF Presentation	cfhd_sets/12.5_25_50/t1	ceac_sets/eac3/at1
9.3 Random Access of a WAVE Presentation	cfhd_sets/12.5_25_50/t1	ceac_sets/eac3/at1
9.4 Splicing of WAVE Program with Baseline Constraints	cfhd_sets/12.5_25_50/splice_main cfhd_sets/12.5_25_50/splice_ad	ceac_sets/eac3/at3 ceac_sets/eac3/at4

12.6 Media Profile: AC-4, Single Stream ('ca4s')

12.6.1 Introduction

The media profile AC-4, Single Stream is defined in [WAVE-CON], clause 4.3 and [ETSI AC-4].

12.6.2 Capability Discovery Options

The generic capability discovery from clause 12.6.1 applies with the following specific aspects:

- `<brand>` is set to `"ca4s"`.
- `<codecs>` is set to one of the following: `"ac-4.02.01.03, ac-4.02.01.02, ac-4.02.01.01 or ac-4.02.01.00"` as defined in [WAVE-CON], Table 2.
- `channels` is defined by the application logic.
- `sampleRate` is defined by the application logic.
- `spatialRendering` should be omitted unless the clients expects to receive a capability indication whether the device can spatially render the material, in which case it should be set to TRUE. If the current output device is an HDMI endpoint, the user agent would only report this configuration as supported if the HDMI endpoint capabilities indicate spatial rendering support. If the current output device are speakers or headphones, the user agent would only report this configuration as supported when virtualization technology is applied.

NOTE: See [DASH] and ETSI TS 103190-2 v1.2.1, Annex E and Annex G for details on how to extract parameters from manifest files or assets.

12.6.3 Source Buffer Initialization Requirements

The general Source Buffer Initialization requirements as defined in clause 12.1.3 apply with codecs set to `"ac-4.02.01.03", "ac-4.02.01.02", "ac-4.02.01.01"` or `"ac-4.02.01.00"`.

12.6.4 Content Options

Encoding and Packaging options

- Codec: Dolby AC-4
- Brands: cmf2, ca4s

- CMAF Fragment durations
  - 2 seconds
- CMAF Switching Set with 2 Tracks of different bitrates

Source Content Options

- Stereo only, 48 kHz

12.6.5 Test Content

The test content for Dolby AC-4 audio is documented in Table 18.

Table 18: Test Content for Enhanced AC-4

ID	PN#	CMAF brand	Codec	brand	Bitrate kbits/s	Sample Rate kHz	Channel	CMAF Fragment durations	Chunks	Duration (s)
at1	PN01	ca4s	ac-4.02.01.00	cmfc, cmf2	64	48	Stereo	2s	1	30
at2	PN01	ca4s	ac-4.02.01.00	cmfc, cmf2	64	48	Stereo	2s	4	30
at3	PN03	ca4s	ac-4.02.01.00	cmfc, cmf2	64	48	Stereo	2s	1	10
at4	PN04	ca4s	ac-4.02.01.00	cmfc, cmf2	64	48	Stereo	2s	1	5.76

12.6.6 Playback Requirements

The playback requirements for AC-4 are as follows:

- Shall support the following playback requirements as documented in clause 8:
  - 8.2 Sequential Track Playback for all prioritized content options
  - 8.3 Random Access to Fragment for all prioritized content options
  - 8.4 Random Access to Time for all prioritized content options
  - 8.5 Switching Set Playback
  - 8.6 Regular Playback of Chunked Content for all prioritized content options
  - 8.7 Regular Playback of Chunked Content, non-aligned append for all prioritized content options
- Should support the following playback requirements as documented in clause 8:
  - 8.8 Playback over WAVE Baseline Splice Constraints
  - 8.9 Out-Of-Order Loading for all prioritized content options
  - 8.10 Overlapping Fragments for all prioritized content options
  - 8.12 Playback of Encrypted Content for all prioritized encrypted content options
  - 8.13 Restricted Splicing of Encrypted Content
  - 8.14 Sequential Playback of Encrypted and Non-Encrypted Baseline Content

12.6.7 Test Cases

The test cases for Dolby AC-4 audio are documented in Table 19, where the content options are defined in section 12.6.5.

Table 19: Test Cases for AC-4 ('ca4s')

Test Code	Content ID
8.2 Sequential Track Playback	at1
8.3 Random Access to Fragment	at1
8.4 Random Access to Time	at1
8.6 Regular Playback of Chunked Content	at2
8.7 Regular Playback of Chunked Content, non-aligned append	at2
8.8 Playback over WAVE Baseline Splice Constraints	at3, at4
8.9 Out-Of-Order Loading	at1
8.10 Overlapping Fragments	at1

If AC-4 audio is tested in a presentation playback, the test cases are documented in Table 20.

Table 20: Test Cases for presentation playback for AC-4 ('ca4s')

Test Code	Video Content ID	Audio Content ID
9.2 Regular Playback of a CMAF Presentation	cfhd_sets/12.5_25_50/t1	ca4s_sets/ac4/at1
9.3 Random Access of a WAVE Presentation	cfhd_sets/12.5_25_50/t1	ca4s_sets/ac4/at1
9.4 Splicing of WAVE Program with Baseline Constraints	cfhd_sets/12.5_25_50/splice_main cfhd_sets/12.5_25_50/splice_ad	ca4s_sets/ac4/at3 ca4s_sets/ac4/at4

12.7 Media Profile: MPEG-H, Single Stream ('cmhs')

12.7.1 Introduction

The media profile MPEG-H, Single Stream is defined in [WAVE-CON], clause 4.3 and [CMAF].

12.7.2 Capability Discovery Options

The generic capability discovery from clause 12.1.2 applies with the following specific aspects:

- `<brand>` is set to `"cmhs"`.
- `<codecs>` is set to one of the following: `"mhm1.0x0B, mhm1.0x0C, mhm1.0x0D"` as defined in [WAVE-CON], Table 2.
- `channels` is defined by the application logic.
- `sampleRate` is defined by the application logic.

12.7.3 Source Buffer Initialization Requirements

The general Source Buffer Initialization requirements as defined in clause 12.1.3 apply with codecs set to `"mhm1.0x0B"`, `"mhm1.0x0C"` or `"mhm1.0x0D"`.

12.7.4 Content Options

This is for further study.

12.7.5 Test Content

This is for further study.

12.7.6 Playback Requirements

This is for further study.

12.7.7 Test Cases

This is for further study.

12.8 Media Profile: DTS-HD Audio ('dts1')

12.8.1 Introduction

The media profile DTS-HD Audio is defined in [WAVE-CON], clause 4.3 and [DTS-HD] Annex H.

12.8.2 Capability Discovery Options

The generic capability discovery from clause 12.1.2 applies with the following specific aspects:

- `<brand>` is set to `"dts1"`
- `<codecs>` is set to one of the following: `"dtsc, dtse"` as defined in [WAVE-CON], Table 2
- `channels` is not used. All channel layouts supported in the `'dts1'` brand can be supported by every client.
- `sampleRate` is not used. Only 48000 Hz is used in the `'dts1'` profile.

12.8.3 Source Buffer Initialization Requirements

The general Source Buffer Initialization requirements as defined in clause 12.1.3 apply with codecs set to `"dtsc"` or `"dtse"`.

12.8.4 Content Options

Encoding and Packaging options

- Codec: DTS Codecs
- CMAF Fragment durations
  - 1.984 seconds

- CMAF Switching Set with 2 Tracks of different bitrates

Source Content Options

- Stereo only, 48 kHz

12.8.5 Test Content

The test content for DTS-HD audio, codec `'dtsc'` is documented in Table 21 and the test content for DTS-HD audio, codec `'dtse'` is documented in Table 22.

NOTE: Greyed columns are not yet implemented.

Table 21: Test Content for DTS-HD audio, codec `'dtsc'`

DTS core test stream	1 (base stream)	1a (alternative)	2 (chunked)	3 (encrypted)	3a (alternative)
CMAF brand	dts1	as 1	as 1	as 1	as 1
Codec	dtsc	as 1	as 1	as 1	as 1
brand	cmf2	as 1	as 1	as 1	as 1
Sample Rate	48kHz	as 1	as 1	as 1	as 1
Channel	Stereo	as 1	as 1	as 1	as 1
CMAF Fragment durations	2.048 seconds	as 1	as 1	as 1	as 1
Chunk per CMAF Fragement	1	as 1	4	as 1	as 1
Encryption	unencrypted	as 1	as 1	Clearkey	Clearkey

Table 22: Test Content for DTS-HD audio, codec `'dtse'`

DTS LBR test stream	1 (base stream)	1a (alternative)	2 (chunked)	3 (encrypted)	3a (alternative)
CMAF brand	dts1	as 1	as 1	as 1	as 1
Codec	dtse	as 1	as 1	as 1	as 1
brand	cmf2	as 1	as 1	as 1	as 1
Sample Rate	48kHz	as 1	as 1	as 1	as 1
Channel	Stereo	as 1	as 1	as 1	as 1
CMAF Fragment durations	2.048 seconds	as 1	as 1	as 1	as 1
Chunk per CMAF Fragement	1	as 1	4	as 1	as 1
Encryption	unencrypted	as 1	as 1	Clearkey	Clearkey

12.8.6 Playback Requirements

This is for further study.

12.8.7 Test Cases

The test cases for DTS-HD audio, codec `'dtsc'` is documented in Table 23 and the test cases for DTS-HD audio, codec `'dtse'` is documented in Table 24.

NOTE: Greyed columns are not yet implemented.

Table 23: Test Cases for DTS-HD audio, codec `'dtsc'`

DTS core test stream	1 (base stream)	1a (alternative)	2 (chunked)	3 (encrypted)	3a (alternative)
8.2 Sequential Track Playback	X				
8.3 Random Access to Fragment	X				
8.4 Random Access to Time	X				
8.5 Switching Set Playback					
8.6 Regular Playback of Chunked Content			X		
8.7 Regular Playback of Chunked Content, non-aligned append			X		
8.8 Playback over WAVE Baseline Splice Constraints	X	X			
8.9 Out-Of-Order Loading	X				
8.10 Overlapping Fragments	X				
8.12 Playback of Encrypted Content				X	
8.13 Restricted Splicing of Encrypted Content				X	X
8.14 Sequential Playback of Encrypted and Non-Encrypted Baseline Content	X			X	
9.2 Regular Playback of a CMAF Presentation	X				
9.3 Random Access of a WAVE Presentation	X				
9.4 Splicing of WAVE Program with Baseline Constraints					

Table 24: Test Cases for DTS-HD audio, codec `'dtse'`

DTS LBR test stream	1 (base stream)	1a (alternative)	2 (chunked)	3 (encrypted)	3a (alternative)
8.2 Sequential Track Playback	X				
8.3 Random Access to Fragment	X				
8.4 Random Access to Time	X				
8.5 Switching Set Playback					
8.6 Regular Playback of Chunked Content			X		
8.7 Regular Playback of Chunked Content, non-aligned append			X		
8.8 Playback over WAVE Baseline Splice Constraints	X	X			

8.9 Out-Of-Order Loading	X				
8.10 Overlapping Fragments	X				
8.12 Playback of Encrypted Content				X	
8.13 Restricted Splicing of Encrypted Content				X	X
8.14 Sequential Playback of Encrypted and Non-Encrypted Baseline Content	X			X	
9.2 Regular Playback of a CMAF Presentation	X				
9.3 Random Access of a WAVE Presentation	X				
9.4 Splicing of WAVE Program with Baseline Constraints					

12.9 Media Profile: USAC Stereo ('casu')

12.9.1 Introduction

The media profile USAC Stereo is defined in [WAVE-CON], clause 4.3 and [CMAF].

12.9.2 Capability Discovery Options

The generic capability discovery from clause 12.1.2 applies with the following specific aspects:

- `<brand>` is set to `"casu"`.
- `<codecs>` is set to `"mp4a.40.42"` as defined in [WAVE-CON], Table 2.
- `channels` is defined by the application logic.
- `sampleRate` is defined by the application logic.

12.9.3 Source Buffer Initialization Requirements

The general Source Buffer Initialization requirements as defined in clause 12.1.3 apply with codecs set to `"mp4a.40.42"`.

12.9.4 Content Options

This is for further study.

12.9.5 Test Content

This is for further study.

12.9.6 Playback Requirements

This is for further study.

12.9.7 Test Cases

This is for further study.

13 SUBTITLE MEDIA PROFILES

13.1 General

The WAVE content specification [WAVE-CON] defines a set of WAVE Media Profiles for subtitles.

The primary focus of CTA WAVE is content streaming and playback by an HTML5 application using MSE and, where needed, EME. During such playback, subtitles and captions are typically decoded and presented by a JavaScript library that is included with the application. For example, for IMSC1, see https://github.com/sandflow/imscJS or https://github.com/Dash-Industry-Forum/dash.js.

Such libraries have dependencies on the HTML5 user agent implementation, such as graphics rendering and synchronization with the video and audio. This synchronization is done by the JavaScript library reading the HTML5 `currentTime` property that is verified as part of WAVE testing. The tests in this specification include observations for the accuracy of the fit between the value of `currentTime` and the currently-output video or audio. These tests are part of WAVE's overall focus on HTML5/MSE/EME testing. Other basic subtitle/captioning functionality in HTML5/MSE/EME playback is not tested by WAVE.

However, a number of resources exist for testing a subtitle decoder; examples for IMSC1/TTML include: https://github.com/w3c/imsc-tests (which is cited in the WAVE Media Profile for IMSC1 Text) and https://dvb.org/specifications/verification-validation/dvb-ttml-subtitling-test-streams/. The W3C publishes IMSC1 test vectors at https://github.com/w3c/imsc-tests (which is cited in the WAVE Media Profile for IMSC1 Image).

WAVE does not currently provide test suite support for native media rendering (the so-called "Type 1").

WAVE collects and develops test content to meet industry needs for streaming interoperability. If an organization has test content suitable for testing decoding and presentation of IMSC1/TTML and wishes to add to the WAVE pool of qualified material, please contact standards@CTA.tech.

North American regulations require support for CTA-608/708 subtitles under certain conditions. The WAVE Content Specification does not require support and the WAVE Device Test Suite does not include any tests for handling CTA-608/708 subtitles. However, the DASH Industry Forum has a .js script for handling CTA-608. It can be found at https://github.com/Dash-Industry-Forum/dash.js/blob/development/externals/cea608-parser.js.

14 OTHER DEVICE PLAYBACK REQUIREMENTS

This version of the specification does not define any additional playback requirements beyond media profile playback. In the future this may for example define requirements on supporting certain graphics overlays, protocol versions of HTTP, etc.

15 DEVICE CORE PROFILES

15.1 Introduction

A device core profile defines a set of playback and other requirements that must be supported by a device to claim conformance against a device core profile.

This version of the specification does not define a device core profile.

16 DEVICE EXTENSION PROFILES

16.1 Introduction

A device extension profile defines a set of playback and other requirements that must be supported by a device to claim conformance against a device extension profile.

A device extension profile assumes that at least one device core profile is supported.

This version of the specification does not define a device core profile.

17 CONFIGURATIONS

17.1 Introduction

Configurations are requirements for devices. A device shall support at least one of any of the defined configurations.

17.2 Encryption

17.2.1 Configuration Options

A WAVE device shall support at least one of the two encryption modes: `"cenc"` or `"cbcs"`.

A WAVE device should support both encryption modes: `"cenc"` and `"cbcs"`.

17.2.2 Capability Discovery

Capability discovery for encryption configuration is for further study.

17.2.3 Playback Requirements

Please refer to clause 7.

Annex A: Device Capability Discovery

A.1 General

The application needs to determine if it can playback the offered content following the requirements of this specification.

It is important to assume that the content is properly labelled (through media profile identifier and other indicators) and formatted according to the controlling specifications, primarily the WAVE content specification [WAVE-CON].

Labeling of the content may for example be done through one or more of the following means:

- WAVE content signaling as defined in the WAVE Content Specification [WAVE-CON].
- The Internet media type of the defined in RFC6381, including the profile and codecs parameter.
- Signaling of the content in the manifest, for example in the DASH-MPD using Adaptation Set signaling, predominantly the @mimeType and @codecs parameter or in HLS M3U8 playlist.
- The signaling in the CMAF Header, specifically:
    - The compatibility brands in the ftyp box, and
    - The information in the sample description box in the CMAF Header.

Note that the information may be duplicated on different levels. For example, the DASH Profile for CMAF content in as defined in ISO/IEC 23009-1 maps CMAF information to the manifest.

For each media profile, the signaling requirements are provided in the WAVE Content Specification. This specification provides means on how to use content signaling for capability discovery.

A.2 Capability Discovery Options

A.2.1 Introduction

This section discusses options that were considered during the development of this specification that may or may not be available or may be available as a proprietary solution or arrangement.

A.2.2 Media Profile

The application uses the media profile for capability discovery. The media profile may for example be provided in the manifest or the CMAF Header in the ftyp box. The application queries the device of the media profile using the `isSupportedType()` API if it can be played back using:

- `<mediatype>/mp4 profiles="<media profile 4CC>"`

The device may provide one of the following answers:

- `Yes`: If yes is provided, then the playback requirements for this media profile as documented in this specification are expected to be fulfilled.
- `No`: If no is provided, then the playback of the media profile is not supported by the device and the application shall not playback this media profile.
- `Unknown`: In this case, the application should find other options to identify if the media profile can be played back.

Note that the media profile signaling does not support the detailed configuration signaling and requires an additional capability mechanism on which configuration is preferably used.

A.2.3 CMAF Header

In this case an API between the app and the platform exists, such that the application queries the device if the content described in the CMAF header can be played back. This has the advantages of being complete, accurate, and future-proof, but the drawback of not being human-readable and possibly requiring the transmission of more information than the other approaches. In addition, there may be cases where not all information is sufficiently provided in the CMAF Header, for example in the case of Inband Parameter Sets.

Again, the device may provide one of the following answers:

- `Yes`: If yes is provided, then the playback requirements for this media profile as documented in this specification are expected to be fulfilled.
- `No`: If no is provided, then the playback of the media profile is not supported by the device and the application shall not playback this media profile.
- `Unknown`: In this case, the application should find other options to identify if the media profile can be played back.

If a no or an unknown is provided, the response should provide an indication based on what feature the device rejected the playback.

A.2.4 MIME Subparameters

This option consists in using one or more MIME sub-parameters to describe the different required capabilities (pre-decoding, decoding, and post-decoding). It is the most commonly-used option today because it has the advantages of enabling progressive, detailed, compact and nearly human-readable signaling.

Post-decoding requirements are indicated in the ISO base media file format with restricted schemes. For example, the 'resv' sample entry type can be used for video tracks that require certain post-decoding operations. Similarly, pre-decoding requirements are indicated in the ISO base media file format with the protected scheme.

In this case, the application uses the detailed MIME type string for communication with the device platform. The application queries the device of the media profile can be played back using:

- `<mediatype>/mp4 mime-subparameters`

The most-commonly used parameter is the `codecs` parameter. For details refer to RFC6381.

Again, the device may provide one of the following answers:

- `Yes`: If yes is provided, then the playback requirements for this media profile as documented in this specification are expected to be fulfilled.

- `No`: If no is provided, then the playback of the media profile is not supported by the device and the application shall not playback this media profile.
- `Unknown`: In this case the application should find other options to identify if the media profile can be played back.

If a no or an unknown is provided, the response should provide an indication based on what feature the device rejected the playback.

A.2.5 Media Capabilities

The Media Capabilities API [W3C MCAP] provides an improved alternative to the `isTypeSupported()API` for determining whether a given user agent is capable of encoding, decoding and rendering a piece of content.

To determine if a user-agent can decode a particular piece of content, the `mediaCapabilities.decodingInfo()` method is called. The method takes an instance of a `MediaDecodingConfiguration` object:

```
dictionary MediaDecodingConfiguration : MediaConfiguration {
    VideoConfiguration video;
    AudioConfiguration audio;
    required MediaDecodingType type;
    MediaCapabilitiesKeySystemConfiguration keySystemConfiguration;
};
```

as an argument and returns as a Promise a `MediaCapabilitiesInfo` object, as defined below:

```
dictionary MediaCapabilitiesInfo {
  required boolean supported;
  required boolean smooth;
  required boolean powerEfficient;
};
```

The Boolean `'supported'` attribute will return `TRUE` if the user agent can decode the supplied audio or video configuration. The two additional Boolean attributes indicate if this can be done in a smooth and power efficient manner, respectively.

For usage within a HTML5 playback environment, the `MediaDecodingType` of the `MediaDecodingConfiguration` instance is set to an enumerated string value of `"media-source"`.

The video and audio configurations as defined in [W3C MCAP] are provided:

```
dictionary VideoConfiguration {
  required DOMString contentType;
  required unsigned long width;
  required unsigned long height;
  required unsigned long long bitrate;
  required double framerate;
  boolean hasAlphaChannel;
  HdrMetadataType hdrMetadataType;
  ColorGamut colorGamut;
  TransferFunction transferFunction;
};
```

```
dictionary AudioConfiguration {
  required DOMString contentType;
  DOMString channels;
  unsigned long long bitrate;
  unsigned long samplerate;
  boolean spatialRendering;
};
```

The required contentType parameter is equivalent to the MIME string described in A.2.3. If the MIME type does not imply a codec, the string shall also have one and only one parameter that is named codecs with a value describing a single media codec. Otherwise, it must contain no parameters. Profile parameters are not allowed.

There are a number of required parameters that must be supplied within a VideoConfiguration or AudioConfiguration. These can typically be extracted by parsing the relevant manifest or playlist. If this is not available, then they can be extracted from a CMAF header (initialization) file. For each media profile, the information is provided in this specification in clause 11 and 12.

In addition, for a MediaDecodingConfiguration to describe [W3C EME], a keySystemConfiguration shall be present.

```
dictionary MediaCapabilitiesKeySystemConfiguration {
  required DOMString keySystem;
  DOMString initDataType = "";
  MediaKeysRequirement distinctiveIdentifier = "optional";
  MediaKeysRequirement persistentState = "optional";
  sequence<DOMString> sessionTypes;
  KeySystemTrackConfiguration audio;
  KeySystemTrackConfiguration video;
};
```

With

- The *keySystem* member represents a keySystem name as described in [W3C EME].
- The *initDataType* member represents a single value from the initDataTypes sequence described in [W3C EME].
- The *distinctiveIdentifier* member represents a distinctiveIdentifier requirement as described in [W3C EME].
- The *persistentState* member represents a persistentState requirement as described in [W3C EME].
- The *sessionTypes* member represents a sequence of required sessionTypes as described in [W3C EME].
- The *audio* member represents a KeySystemTrackConfiguration associated with the AudioConfiguration.
- The *video* member represents a KeySystemTrackConfiguration associated with the VideoConfiguration.

```
dictionary KeySystemTrackConfiguration {
  DOMString robustness = "";
  DOMString? encryptionScheme = null;
};
```

With

- The *robustness* member represents a robustness level as described in [W3C EME].
- The *encryptionScheme* member represents an encryptionScheme as described in [W3C EME].

Well-known values for encryptionScheme are:

- ***cenc***: The "cenc" mode, defined in [CENC], section 4.2a. AES-CTR mode full sample and video NAL subsample encryption.
- ***cbcs***: The "cbcs" mode, defined in [CENC], section 4.2d. AES-CBC mode partial video NAL pattern encryption. For video, the spec allows various encryption patterns.
- ***cbcs-1-9***: The same as "cbcs" mode, but with a specific encrypt:skip pattern of 1:9 for video, as recommended in [CENC], section 10.4.2.

Annex B: Relevant HTML-5 APIs (Informative)

B.1 General

This clause documents the HTML-5 APIs that are relevant to the implementation of the high-level tests as defined in this specification.

B.2 Relevant Web Media APIs

The following APIs are relevant, and their existence is assumed when implementing the tests based on an HTML5 platform. The exact mapping will be addressed in a future version of this specification.

1) **HTMLMediaElement**
    - Properties
        i. HTMLMediaElement.buffered - OBSERVATION - returns a TimeRanges object that indicates the ranges of the media source that the browser has buffered (if any) at the moment the buffered property is accessed.
        ii. HTMLMediaElement.`currentTime` - OBSERVATION INPUT - is a double indicating the current playback time in seconds. Setting this value seeks the media to the new time.
    - Methods
        i. HTMLMediaElement.fastSeek() - INPUT - directly seeks to the given time.
        ii. HTMLMediaElement.pause() - INPUT - pauses the media playback.
        iii. HTMLMediaElement.play() - INPUT - begins playback of the media.

Other properties and methods are available as follows:

2) **HTMLMediaElement**
    - Properties
        i. HTMLMediaElement.defaultPlaybackRate - OBSERVATION INPUT - Is a double indicating the default playback rate for the media.
        ii. HTMLMediaElement.duration - OBSERVATION - Returns a double indicating the length of the media in seconds, or 0 if no media data is available.
        iii. HTMLMediaElement.initialTime - OBSERVATION - Returns a double that indicates the initial playback position in seconds.
        iv. HTMLMediaElement.mediaKeys - OBSERVATION INPUT - Returns a MediaKeys object or null. MediaKeys is a set of keys that an associated HTMLMediaElement can use for decryption of media data during playback.
        v. HTMLMediaElement.muted - OBSERVATION INPUT - Is a Boolean that determines whether audio is muted. true if the audio is muted and false otherwise.
        vi. HTMLMediaElement.paused - OBSERVATION - Returns a Boolean that indicates whether the media element is paused.
        vii. HTMLMediaElement.playbackRate - Is a double that indicates the rate at which the media is being played back.
        viii. HTMLMediaElement.played - OBSERVATION - Returns a TimeRanges object that contains the ranges of the media source that the browser has played, if any.
        ix. HTMLMediaElement.preservesPitch - OBSERVATION INPUT - Is a Boolean that determines if the pitch of the sound will be preserved.
        x. HTMLMediaElement.seekable - OBSERVATION - Returns a TimeRanges object that contains the time ranges that the user is able to seek to, if any.

xi. HTMLMediaElement.volume - OBSERVATION INPUT - Is a double indicating the audio volume, from 0.0 (silent) to 1.0 (loudest).

xii. HTMLVideoElement.height - OBSERVATION INPUT - Is a DOMString that reflects the height HTML attribute, which specifies the height of the display area, in CSS pixels.

xiii. HTMLVideoElement.width - OBSERVATION INPUT - Is a DOMString that reflects the width HTML attribute, which specifies the width of the display area, in CSS pixels.

xiv. HTMLVideoElement.videoHeight - OBSERVATION - Returns an unsigned long containing the intrinsic height of the resource in CSS pixels, taking into account the dimensions, aspect ratio, clean aperture, resolution, and so forth, as defined for the format used by the resource.

xv. HTMLVideoElement.videoWidth - OBSERVATION - Returns an unsigned long containing the intrinsic width of the resource in CSS pixels, taking into account the dimensions, aspect ratio, clean aperture, resolution, and so forth, as defined for the format used by the resource.

3) `MediaSource`

- Properties

  i. `MediaSource.SourceBuffers` – OBSERVATION - Returns a `SourceBuffer`List object containing the list of `SourceBuffer` objects associated with this `MediaSource`.

  ii. `MediaSource`.active`SourceBuffers` - OBSERVATION - Returns a `SourceBuffer`List object containing a subset of the `SourceBuffer` objects contained within `SourceBuffers` — the list of objects providing the selected video track, enabled audio tracks, and shown/hidden text tracks.

  iii. `MediaSource`.readyState – OBSERVATION - Returns an enum representing the state of the current `MediaSource`, whether it is not currently attached to a media element (closed), attached and ready to receive `SourceBuffer` objects (open), or attached but the stream has been ended via `MediaSource`.endOfStream().

  iv. `MediaSource`.duration – OBSERVATION INPUT - The duration of the media that is playing

- Methods

  i. `MediaSource`.add`SourceBuffer`() - INPUT - Creates a new `SourceBuffer` of the given MIME type and adds it to the `MediaSource`'s `SourceBuffers` list.

  ii. `MediaSource`.remove`SourceBuffer`() – INPUT - Removes the given `SourceBuffer` from the `SourceBuffers` list associated with this `MediaSource` object.

  iii. `MediaSource`.endOfStream() – INPUT - Signals the end of stream.

Annex C: Test Content Format Specification (Normative)

C.0 Introduction

This Annex documents the annotations and other required details of the WAVE Project "Mezzanine Content" and "Encoded Content", excluding details of the encoding itself. Mezzanine Content is the term used for the original source content (such as "Big Buck Bunny", "Tears of Steel", or "Croatia") after it has been annotated with artifacts intended for test automation. These artifacts may be audio markers, QR codes, frame markers or the like. Once the Mezzanine Content is ready, it is encoded in a valid WAVE Media Profile to become Encoded Content.

The Mezzanine Content is intended to be processed by a script to generate Encoded Content (actual test vectors). This script may rely on other details of the Mezzanine Content, such as file name format. In addition, actual tests that rely on Encoded Content also have certain expectations such as file name format.

This specification is intended to document these details sufficiently to allow a developer to create valid Mezzanine Content in the future, and for a developer to process Encoded Content based on valid Mezzanine Content, where 'valid' means that the generating script can receive the new content as input and generate Encoded Content as output based on the requirements in this specification.

C.1 References

C.1.1 Normative References

The following documents, in whole or in part, are normatively referenced in this Annex and are indispensable for its application. For dated references, only the dated edition applies. For undated references, the latest edition of the referenced document (including any amendments and corrigenda) applies.

[1] CTA-5001-E, *Web Application Video Ecosystem – Content Specification*, December 2022, https://cta.tech/standards.

[2] CTA-5003-A, *Wave Application Video Ecosystem – Device Playback Capabilities*, https://cta.tech/standards.

C.1.2 Informative References

The following documents contain provisions that, through reference in this text, constitute informative provisions of this document. At the time of publication, the editions indicated were valid. All documents are subject to revision, and parties to agreements based on this document are encouraged to investigate the possibility of applying the most recent editions of the documents listed here.

[3] Cousine Font, https://fonts.google.com/specimen/Cousine.

[4] RFC 5234, *Augmented BNF for Syntax Specifications: ABNF*, January 2008, https://datatracker.ietf.org/doc/html/rfc5234.

[5] ffmpeg Documentation, Codec Options, https://ffmpeg.org/ffmpeg-all.html#Codec-Options.

C.1.3 Document Notation and Conventions

The following terms are used to specify conformance elements of this specification. These are adopted from the ISO/IEC Directives, Part 2, Annex H [ISO-P2H ISO-P2H]. For more information, please refer to those directives.

- SHALL and SHALL NOT indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.
- SHOULD and SHOULD NOT indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required, or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.
- MAY and NEED NOT indicate a course of action permissible within the limits of the document.

Terms defined to have a specific meaning within this specification will be capitalized, e.g., "Track", and should be interpreted with their general meaning if not capitalized.

C.1.4 Definitions

For the purposes of this document the following terms and definitions apply.

Term	**Definition**
Encoded Content	When capitalized as Encoded Content, this term means Mezzanine Content that has been encoded in a valid Media Profile according to requirements in this specification.
Media Profile	A WAVE Content Specification Media Profile.
Mezzanine Content	Audio, video or subtitle track data from various source material(s) that has been annotated according to this specification.

C.2 Mezzanine Content Requirements (Normative)

C.2.1 Introduction

To enable testing of device playback capabilities defined in CTA-5003 [2], the audio and video stimulus must consist of audio/video media content with specific annotations, defined in this section. Mezzanine content is prepared with these annotations, which can then be used to encode test content for the different WAVE Media Profiles defined in CTA-5001 [1].

C.2.2 Source Content

To create annotated mezzanine content, source audio and video content is required, to which the annotations will be added. It is preferable for the source audio to be the associated audio track to the source video, as this ensures test content is closer to real-world content and provides additional testing possibilities.

Annotations can be added to any source video content, as the resolution, frame rate and dynamic range can be converted as required during the mezzanine content preparation. Audio annotations may be sounds for human monitoring or computer-detectable watermarks.

In order to create mezzanine video content with the optimal quality, it is preferable to start from source content that has the following properties:

- A resolution equal to or higher than that of the highest resolution needed for testing WAVE Media Profiles.
- A native frame rate equal to that used for testing WAVE Media Profiles or from the same frame rate family (e.g., 15/30/60).
- A dynamic range equivalent to that used for testing WAVE Media Profiles.

Consequently, it is preferable to use multiple different source videos with the appropriate native properties (e.g., 25fps vs 30fps, SDR vs HDR), to generate corresponding sets of mezzanine content.

Audio content should use a WAVE-supported sample rate (currently only 48 kHz), and, where watermarks are used, relatively limited dynamic range.

C.2.3 Human Readable Annotations

C.2.3.1 General

The following human readable annotations shall be added to each frame of source video content:

- label,
- timecode,
- frame number,
- frame rate.

The timecode, frame number and frame rate shall be updated for each video frame. The label shall be fixed for the complete duration of each annotated mezzanine content file.

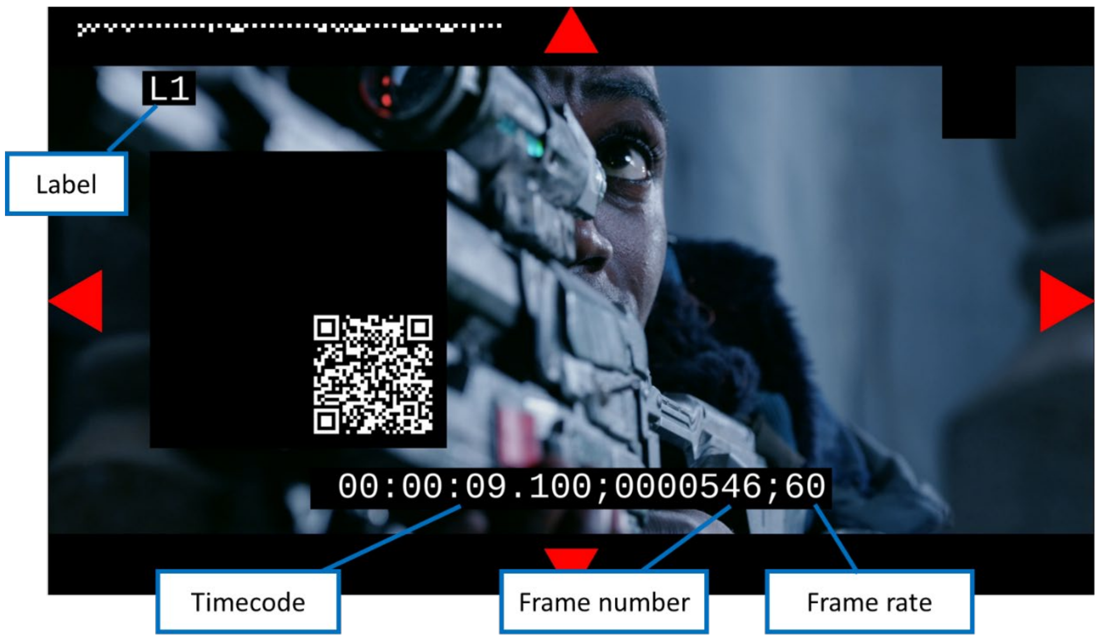


Figure 9: Example of human readable annotations

These properties are used in the following sections, and are defined as follows:

- video_width = the horizontal resolution, in pixels, of the annotated mezzanine content.
- video_height = the vertical resolution, in pixels, of the annotated mezzanine content.
- text_width = the width, in pixels, of a text string rendered on a single line in the annotated mezzanine content.
- text_line_height = the height, in pixels, of a single line of text rendered in the annotated mezzanine content.

Text strings shall be rendered with the following properties:

- font = Cousine Regular [3]
- font color = white
- font size = 0.06 * video_height
- border = 10 pixels
- border color = black

C.2.3.2 Label

The label is used to identify a particular piece of test content during testing, and shall be in the following format expressed in ABNF [4]:

```
label = ALPHA DIGIT     ; e.g. "L2"
```

The label is placed at the top-left of the video. The coordinates of the top-left corner of the label shall be defined as follows:

- x = (video_width – text_width) / 10
- y = 3 * text_line_height

C.2.3.3 Timecode, frame number and frame rate

The timecode, frame number and frame rate shall be grouped together in a single text string, expressed in ABNF [4] as follows:

```
annotation = timestamp ";" framecounter ";" framerate ; e.g.,
00:00:59.417;0003565;60
timestamp = hours ":" minutes ":" seconds "." milliseconds  ; i.e.
HH:MM:SS.mmm
hours = 2DIGIT                                        ; i.e. 00-99
minutes = (%x30-35) DIGIT                             ; i.e. 00-59
seconds = (%x30-35) DIGIT                             ; i.e. 00-59
milliseconds = 3DIGIT                                 ; i.e. 000-999
framecounter = 7DIGIT                                 ; i.e. 0000000-
9999999
framerate = 2*3DIGIT ["." 1*3DIGIT]
```

Fractional frame rates shall be rounded to three decimal places with any trailing zeros truncated – e.g., 23.976, 29.97, 59.94, 119.88.

The text string containing the timecode, frame number and frame rate is placed at the bottom-center of the video. The coordinates of the top-left corner of the text string shall be defined as follows:

- x = (video_width - text_width) / 2
- y = video_height - (4 * text_line_height)

C.2.4 QR Codes

A QR code encoding the text strings from the human readable annotations shall be added to each frame of source video content.

Each QR code shall contain a text string, expressed in ABNF [4] as follows:

```
qrcode_text = label ";" timestamp ";" framecounter ";" framerate
                    ; e.g. L2;00:00:59.417;0003565;60
```

The label, timestamp, framecounter and framerate are defined in the remainder of this Annex.

Each QR code is placed over the left side of the video, at a position that alternates counter-clockwise between four different positions with each successive frame:

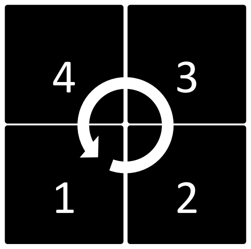


Figure 10: QR code successive positions from frame 1 to 4

The size and coordinates of the top-left corner of each QR code shall be defined as follows, based on the frame number:

qrcode_width = qrcode_height = video_height * 0.25

$$x = \begin{cases} (\text{video\_width} * 0.1), & 0 \leq \text{frame\_number} \% 4 \leq 1 \\ (\text{video\_width} * 0.1) + \text{qrcode\_width}, & 2 \leq \text{frame\_number} \% 4 \leq 3 \end{cases}$$

$$y = \begin{cases} \left(\frac{\text{video\_height}}{2}\right) - \text{qrcode\_width}, & 0 \leq \text{frame\_number} \% 4 < 1 \\ \left(\frac{\text{video\_height}}{2}\right), & 1 \leq \text{frame\_number} \% 4 \leq 2 \\ \left(\frac{\text{video\_height}}{2}\right) - \text{qrcode\_width}, & 2 < \text{frame\_number} \% 4 \leq 3 \end{cases}$$

A black background is inserted behind the QR codes and in front of the source video content to facilitate QR code detection. The position and dimensions of the background shall be as follows:

- x = (video_width * 0.1) - 4
- y = (video_height / 2) + qrcode_width - 4
- width = height = (qrcode_width * 2) + 4

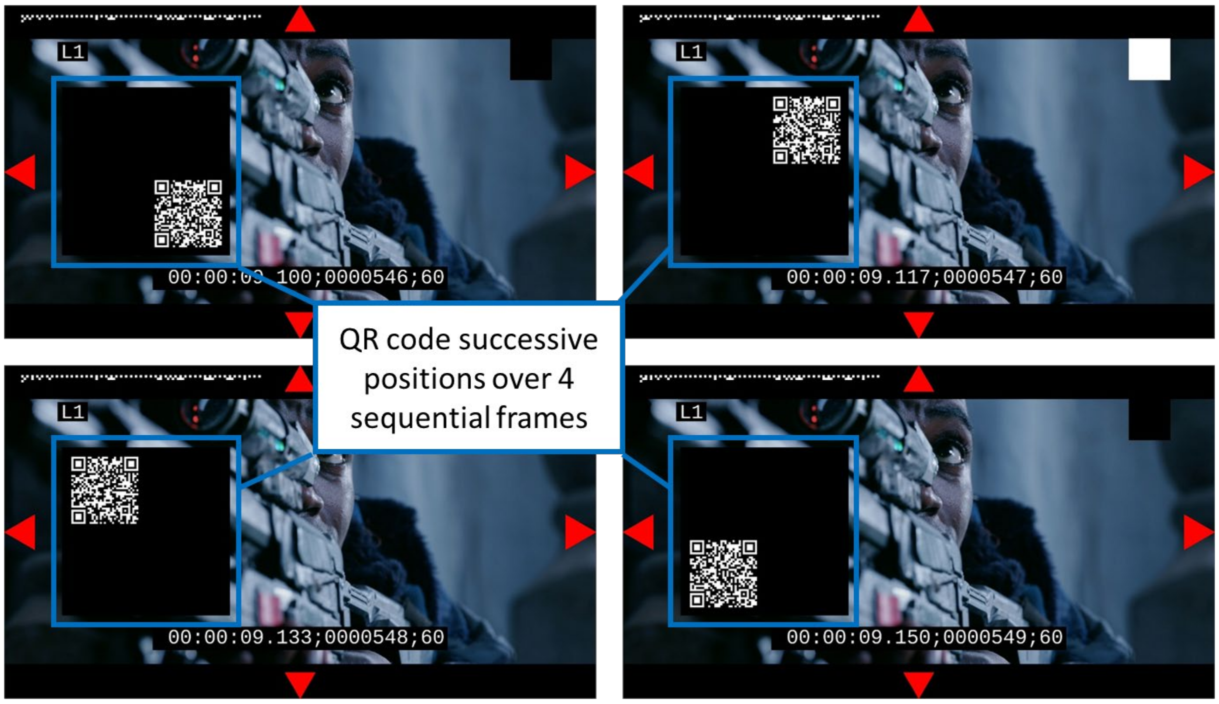


Figure 11: Example of QR codes in 4 successive frames

C.2.5 Audio Sync Marks

A visual audio sync mark shall be added to the top-right of each frame of source video content. This mark turns white (a "flash") for the duration of the corresponding audio feedback (a "beep"), which shall be mixed with the source audio content, to enable testing of audio/video synchronization.

The size and coordinates of the top-left corner of the audio sync mark shall be defined as follows:

av_sync_mark_width = av_sync_mark_height = video_height * 0.125

x = video_width * 0.925 - av_sync_mark_width

y = video_height * 0.1

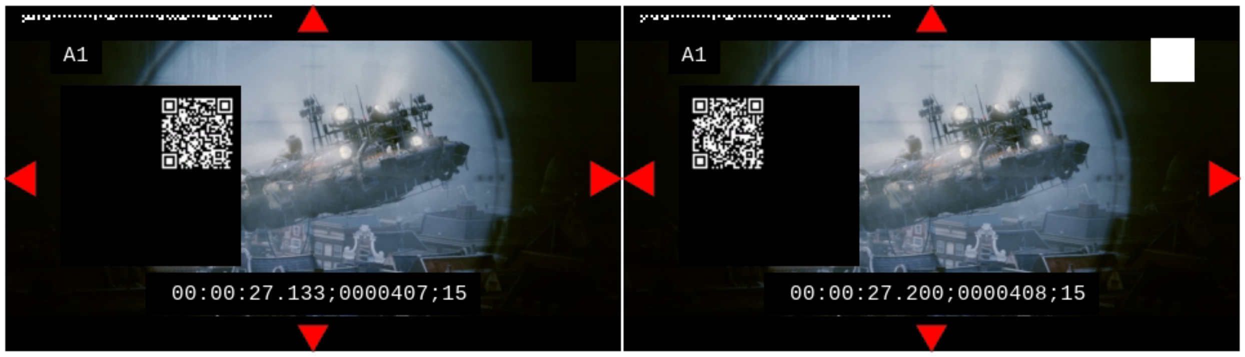


Figure 12: Examples of the audio sync mark indication in the absence of audio feedback (left) and during audio feedback (right)

C.2.6 Audio Watermarks

C.2.6.1 Source Files

Four watermark (pseudo-random noise or pseudo-noise, PN) source files are listed below. The files were generated by a Python program, PNfiles.py, which uses the "seed source string" to initialize a PCG64 bit generator; it then puts one bit per audio sample (at 48000 Hz sample rate) and low-pass filters the result to 7 kHz. While the files may be generated dynamically, Python does not guarantee repeatability of the PCG64 method over library version updates. Therefore the files are generated once and the archived copies are used in the test suite.

Seed Source String	Seed	Use	MD5 Hash	Duration
PN01.wav	80078048049	Primary content, production code	622b1a78a0a7479433645f7b9fa579ec	30
PN02.wav	80078048050	Secondary content, production code	ebb1db88ce70605798f1114e5142e3ef	30
PN03.wav	80078048051	Main (spliced) content	52cd6742edefd9ba5af9565cf32d1824	60
PN04.wav	80078048052	Ad (spliced) content	f44fa674c388e529eafaf7ffa1a1d519	60

In addition, for development purposes, an additional "music plus PN" file is included with the PN archive:

Seed Source String	Seed	Use	MD5 Hash	Duration
AdagioPN01-12dB.wav	n/a	Development purposes	A6C14824708CB8A96FE91D1F2E838F84	60

C.2.6.2 Use of Watermarks

Segments of the PNxx files as short as 20 mS can be detected with clean (no added noise) reception of the audio. This corresponds to the playout device (device under test) connected to the Observation Framework computer via *line-out→direct cable→line-in* (besides analog line-in and line-out ports, USB Audio endpoints on the devices may also be used).

Detecting a PNxx segment in received audio and correlating the segment to its location in the original PNxx recording yields a measurement of `mediaTime`.

C.2.7 Low Density Bit Pattern

A low density bit pattern (LDBP) shall be added to the top-left of each frame of source video content. This LDBP supplements, but does not replace, the QR codes. The QR codes are designed for black-box testing of devices, using a camera to capture device video output, without requiring manufacturer involvement. However, there are also cases where a manufacturer may want to do more in-depth white-box testing, and the LDBP makes the test content more useful for such testing. Processing of the LDBP image is easier to automate for white-box testing than a QR code which changes position in each successive frame. With white-box testing, error correction is not necessary, as video data is processed directly within a device.

Each "bit" in the LDBP shall consist of a 2x2 pixel square for a video resolution of 480x270**,** the lowest resolution tested.

For annotated mezzanine video resolutions higher than 480x270, the bit pattern shall scale linearly with the video resolution. The position and size of the LDBP bits relative to the video shall remain the same.

The LDBP shall contain six lines and each line shall be composed of 106 bits.

The composition of the pattern shall be, in order from top to bottom:

- 2 lines of bits set to 0b0.
- 1 calibration line:
  - Starting with 5 bits set to 0b0.
  - Followed by a sequence of 98 bits alternating between 0b1 and 0b0, starting with 0b1.
  - Ending with 3 bits set to 0b0.
- 2 data lines:
  - Starting with 5 bits set to 0b0.
  - Followed by 2 calibration bits (each the inverse of the bit directly above).
  - Followed by 96 bits of data payload.
  - Ending with 3 bits set to 0b0.

- 1 line of bits set to 0b0.

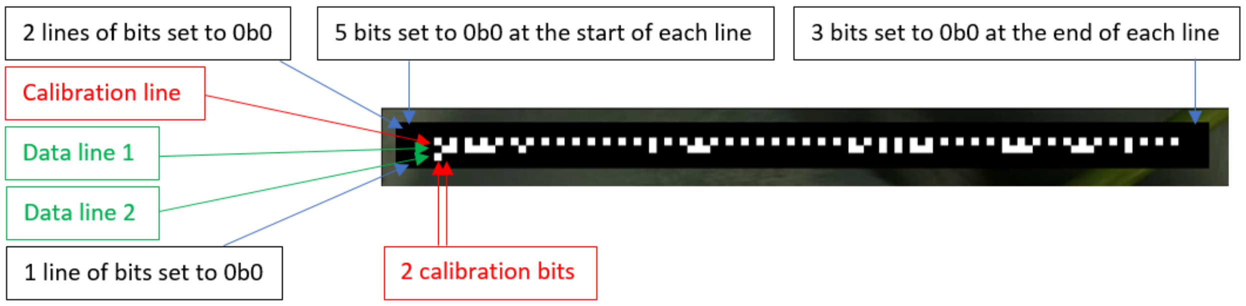

2 lines of bits set to 0b0
5 bits set to 0b0 at the start of each line
3 bits set to 0b0 at the end of each line
Calibration line
Data line 1
Data line 2
1 line of bits set to 0b0
2 calibration bits

Figure 13: Low Density Bit Pattern structure

The data payload shall be encoded on the two data lines in the 2x96 bits following the calibration bits as follows. In order (left to right, top to bottom), each value starting with the LSB, ending with the MSB:

- current frame number (24 bit),
- total number of frames in annotated mezzanine content (24 bit),
- frame rate * 1000 (17 bit),
- horizontal resolution (13 bit),
- vertical resolution (13 bit).

The size and coordinates of the top-left corner of the LDBP shall be defined as follows:

- LDBP_horizontal_pixels = 106
- LDBP_vertical_pixels = 6
- LDBP_width = LDBP_horizontal_pixels / (240 / video_width)
- LDBP_height = LDBP_vertical_pixels / (135 / video_height)
- x = 4 / 480 * video_width
- y = 4/ 270 * video_height

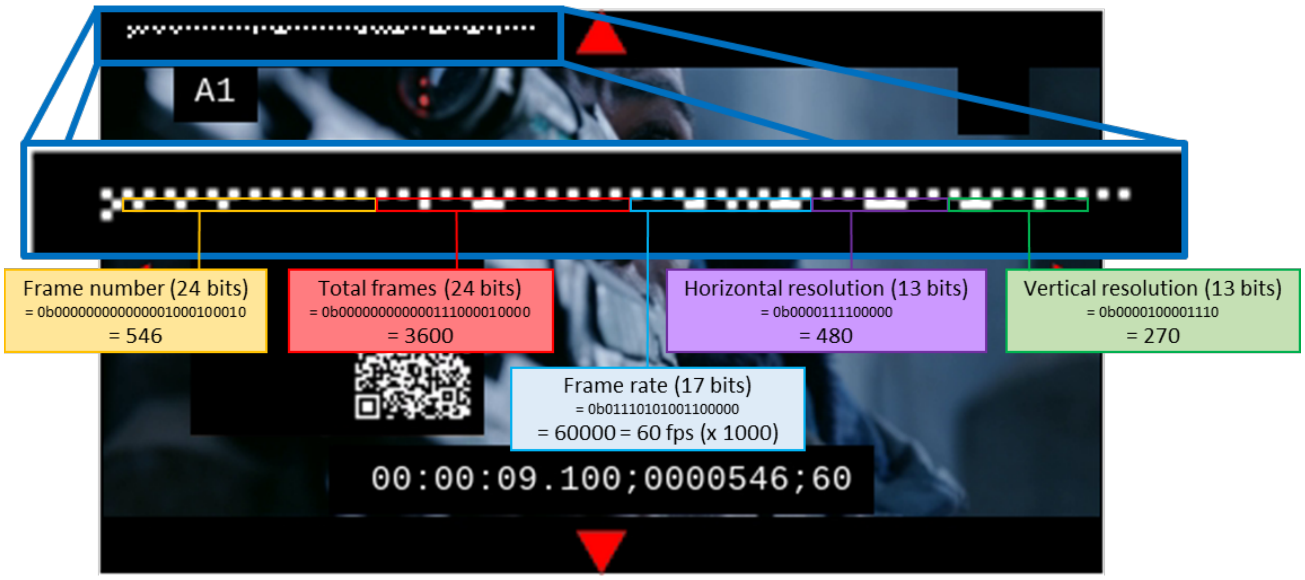


Figure 14: Example of LDBP

C.2.8 Border Indications

Multiple indications shall be added to each frame of source video content, to aid detection of test content edges:

- A two pixel-wide border shall be overlaid on the source video content, with the outer pixel border colored in white and the inner pixel border colored in black.
- Red triangular indicators shall be placed at the center of each edge of the video, pointing outwards, with the summit of each triangle touching the video edge (and therefore obscuring a small portion of the two pixel-wide border).

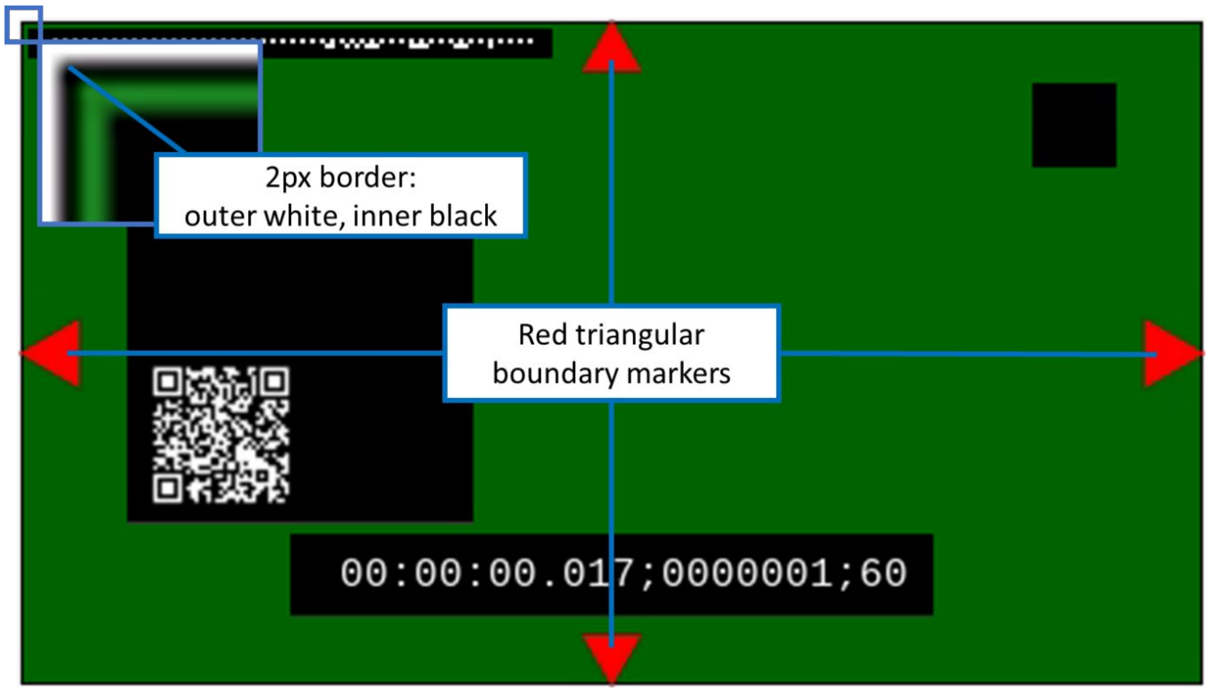


Figure 15: Example of border indications

C.2.9 Start/End Frame Indications

To provide a clear visual indication of the start and end of the test content, the first and last frames of mezzanine content shall be filled with a single color instead of using the corresponding source video content frames.

The first/start frame shall be colored green 8-bit RGB [0,100,0] and the last/end frame red 8-bit RGB [139,0,0].

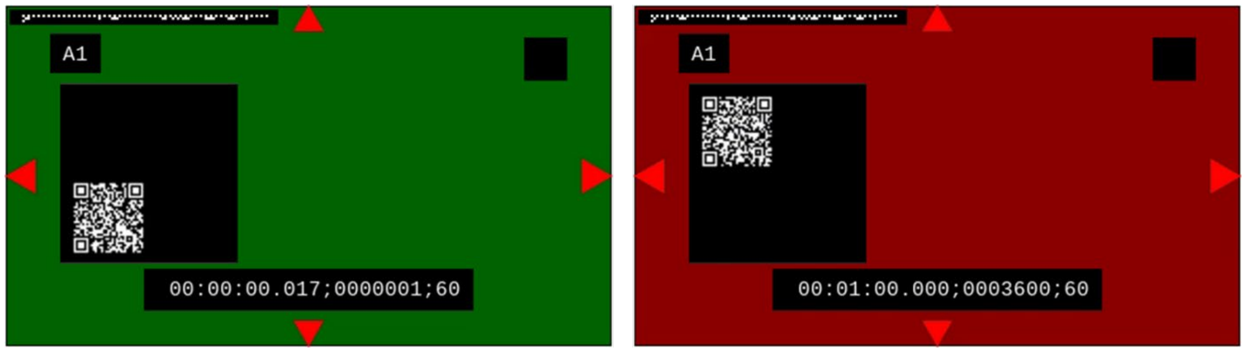


Figure 16: Example of start and end frame indications

C.2.10 Encoding

Mezzanine content shall be encoded using ffmpeg, using one of two profiles depending on the dynamic range and colorspace:

- SDR/BT.709 content shall be encoded with the following parameters:
    - -c:v libx264 -preset slower -crf 5
- HDR/BT.2020 content shall be encoded with the following parameters:
    - -c:v libx265 -preset slower -crf 5
- Audio shall be encoded with the following parameters:
    - -c:a aac -b:a 320k -ac 2

NOTE: While these encodings are sufficiently high quality as to not result in any meaningful loss in quality, future improvements to the mezzanine preparation process may use a more appropriate mezzanine video codec or raw YUV and WAV video and audio data.

C.2.11 File Name Format

Mezzanine content files shall be named as follows:

<prefix>_<label>_<WxH>@<fps>_<duration>.mp4

Where:

- *prefix* is a short string defined based on the source content used – e.g., "tos" for Tears of Steel,
- *label* is the label defined in section C.2.3.2,
- *W* is the mezzanine content width in pixels,
- *H* is the mezzanine content height in pixels,
- *fps* is the mezzanine content frame rate,
- *duration* is the mezzanine content duration in seconds.

Examples:

- tos_L1_1920x1080@59.94_60.mp4
- croatia_I2_1024x576@25_60.mp4

C.2.12 Metadata

C.2.12.0 Overview

Metadata shall be generated together with the mezzanine content and saved to JSON files.

For each mezzanine content file, two JSON files shall be generated:

- Mezzanine content metadata: <mezzanine_file_name>.json

- A/V sync metadata: <mezzanine_file_name>_avsync.json

C.2.12.1 Mezzanine content metadata

The JSON metadata related to the mezzanine content, encoding, file and source file shall include the following information:

- *Mezzanine*:
    - *name* – annotated mezzanine file name.
    - *URI* – mezzanine file location relative to the JSON metadata file.
    - *version* – the official mezzanine release version this file belongs to (0 if none).
    - *specification_version* – the version of the present specification that applies to the annotated mezzanine content.
    - *creation_date* – date the mezzanine content file was created in ISO 8601 YYYY-MM-DD format.
    - *license* – the license text that applies to the annotated mezzanine content.
    - *command_line* – the command line used to call the mezzanine.py script that creates the mezzanine content.
    - *ffmpeg_command_line* – the ffmpeg command line used to encode the mezzanine content.
    - *md5* – MD5 hash of the mezzanine content file.
    - *properties* – information about the video encoding:
        - *width* – mezzanine content video width in pixels.
        - *height* – mezzanine content video height in pixels.
        - *frame_rate* – rounded to 3 decimal places, trailing zeros truncated.
        - *scan* – always progressive for WAVE test content.
        - *pixel_format* – name of the pixel format (e.g., yuv420p).
        - *bit_depth* – bit depth.
        - *color_primaries* – name of the corresponding standard as documented for the color_primaries option in ffmpeg Codec Options [5] (e.g., bt709 for ITU-R BT.709).
        - *transfer_characteristics* – name of the corresponding standard as documented for the color_trc option in ffmpeg Codec Options [5] (e.g., smpte2084 for SMPTE ST 2084).
        - *matrix_coefficients* – name of the corresponding standard as documented for the colorspace option in ffmpeg Codec Options [5], where NC and NCL refer to non-constant luminance and CL to constant luminance (e.g., bt2020nc for ITU-R BT.2020 non-constant luminance).
        - *range* – video range, limited or full.
        - *duration* – in seconds.
        - *frame_count* – total number of frames in the mezzanine content.
        - *start_frame* – frame count used for the first frame of content (expected value: 1).
        - *start_indicator* – presence of a green frame indicating the start of the test content as defined in section C.2.8 (expected value: true).
        - *end_indicator* – presence of a red frame indicating the end of the test content as defined in section C.2.8 (expected value: true).
        - *qr_positions* – number of different positions the QR codes alternate between as defined in section C.2.4 (expected value: 4).
        - *label* – label associated with this mezzanine content as defined in section C.2.3.2.
        - *codec* – short name of the video codec used (e.g., `"H.264"`).
    - source – information about the source content file:
        - *name* – source content file name.
        - *URI* – source content file location.
        - *license* – the license text that applies to the source content.

C.2.12.2 A/V sync metadata

The JSON metadata related to the audio sync marks and beeps defined in section C.2.5 includes the following information:

- *size* – Width and height of the audio sync mark in pixels when generated but should be disregarded as the mark is resized when overlaid on the source video content, as described in section C.2.2.
- *fps* – Frame rate used to generate the audio sync marks and beeps, the same as the mezzanine content frame rate.
- *durationSecs* – Duration (in seconds) audio sync marks and beeps are generated for, the same as the mezzanine content duration.
- *patternWindowLength* – Defines the window of time during which the sequence of audio sync marks and beeps is unique (expected value: 6, the unique sequence repeats after 2^6-1 = 63s).
- *eventCentreTimes* – The timing of each audio sync mark and beep, which start at:
    - eventCentreTime[n] - approxFlashDurationSecs/2,
    - and eventCentreTime[n] - approxBeepDurationSecs /2 respectively.
- *approxBeepDurationSecs* – The duration for which each beep lasts.
- *approxFlashDurationSecs* – The duration for which each audio sync mark is displayed.

C.2.13 Mezzanine Availability

WAVE Mezzanine content releases, including the most recent release, can be found at https://dash-large-files.akamaized.net/WAVE/Mezzanine/releases/

C.3 Encoded Content File Requirements (Normative)

C.3.1 File Name Format

To make the intent and content of file in an Encoded Content suite clear, the file (test vector) name is built from the Media Profile, frame rate family, and stream identifier as follows:

<media_type>_sets/<sub_media_type (frame_rate_family|audio_codec)>/<stream_id>/<upload_date>

- *media_type* – The Media Profile followed by the suffix "_sets".
- *sub_media_type* – A custom identifier for identifying the family of tests. For video this shall be the frame_rate family (e.g., "15_30_60"). For audio this shall be audio codec (e.g., "aac_lc").
- *stream_id* – The identifier for the stream contained in the subfolder as provided by the requirements for generating streams (e.g., "ss1" or "at1").*upload_date* – The date of generation of the stream content.

Examples:

avc_sets/15_30_60/ss1/2021-10-22/ caac_sets/aac_lc/at1/2021-12-04/ caaa_sets/he_aac_v2/at1/2021-12-04/

C.4 Encoded Content Manifest Requirements (Normative)

C.4.1 Introduction

This section includes the requirements that apply to the manifest or MPD of the Encoded Content.

C.4.2 Encoded Content Format Tag

The version of this specification used in the Mezzanine Content and the Encoded Content is tracked in the Asset Identifier of the test content MPD.

As the current version of this specification is 0.1, the tag shall be:

```
<AssetIdentifier schemeIdUri="urn:cta:org:wave-test-mezzanine:unique-id"
value="0.1"/>
```

C.4.3 Encoded Content Options Tags

This section covers tags for information on content options that are not already in the manifest, as JP points out in issue 34 in the Test-Content-Generation repo (with/without picture timing SEI message; with/without VUI timing info; etc.)

C.5 Test Content Availability

WAVE test content releases, including the most recent release, can be found at https://dash-large-files.akamaized.net/WAVE/vectors/releases/.

January 17, 2024

To: Recipients of CTA-5003-A, Device Playback Capabilities

An update to the original publication of CTA-5003-A has been made to change the maximum permitted startup delay (`TSMax`) from 120ms to 1000ms. The attached pages indicate the six places where this change is made (in sections 8.2.3, 8.3.3, 8.4.3, 8.5.3, 8.7.3, and 8.18.3).

Sincerely,

Technology & Standards Staff

Consumer Technology Association

standards@cta.tech

===== 1. CHANGE =====

8.2.3 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback. The recommended value is 1 second.
2) `TSMax`: The maximum permitted startup delay set to 1000ms.

NOTE: This constraint is defined as a first approach but may be refined after running some initial tests.

===== 2. CHANGE =====

8.3.3 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback. The recommended value is 1 second.
2) `TSMax`: The maximum permitted startup delay set to 1000ms.

NOTE: This constraint is defined as a first approach but may be refined after running some initial tests.

===== 3. CHANGE =====

8.4.3 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback.
2) `random_access_time`: Defines the presentation time at which the track is randomly accessed.
3) `TSMax`: The maximum permitted startup delay set to 1000ms.

NOTE: This constraint is defined as the first approach but may be refined after running some initial tests.

===== 4. CHANGE =====

8.5.3 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback.

2) `playout[i]`: Provides the CMAF track number for every fragment position `i=1,…,N`. The value shall be between 1 and K.

   - proposed playout `1, 2, …, K, K-1, …, 1, K, 1, K`, then repeat.

3) `TSMax`: The maximum permitted startup delay is set to 1000ms.

   NOTE: This constraint is defined as a first approach but may be refined after running some initial tests.

===== 5. CHANGE =====

8.7.3 Parameters and Variants

The playback has the following parameters:

1) `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback. This value shall be smaller than `df[k,i]` of all Fragments.

2) `TSMax`: The maximum permitted startup delay set to 1000ms.

NOTE: This constraint is defined as a first approach but may be refined after running some initial tests.

===== 6. CHANGE =====

8.18.3 Parameters and Variants

- `min_buffer_duration`: Expresses the minimum buffer that the Source Buffer maintains in the playback.
- `playout[i]`: Provides the triple (Switching Set, CMAF track number, Fragment number) for every fragment in the first presentation.
- `second_playout_switching_time`: Time to switch from the first to the second presentation. This shall be less than the duration of the first presentation. (This prevents the device entering a waiting state).
- `second_playout[i]`: Provides the triple (Switching Set, CMAF track number, Fragment number) for every fragment in the second presentation.
- `TSMax`: The maximum permitted startup delay of the second presentation, set to 1000ms.

NOTE: TSMAx in this case refers to the startup of the *second* presentation in contrast to the other cases where it typically refers to the initial startup. The value is set to identical values of initial startup.

Consumer Technology Association Document Improvement Proposal

If in the review or use of this document a potential change is made evident for safety, health or technical reasons, please email your reason/rationale for the recommended change to standards@CTA.tech.

Consumer Technology Association
Technology & Standards Department
1919 S Eads Street, Arlington, VA 22202
FAX: (703) 907-7693 standards@CTA.tech

Consumer
Technology
Association™