 ISO/IEC JTC 1/SC 29/WG 03 N0012

**ISO/IEC JTC 1/SC 29/WG 03  
MPEG Systems   
Convenorship: KATS (Korea, Republic of)**

**Document type:** Output Document

**Title:** Advance signalling of MPEG containers content

**Status:** Approved

**Date of document:** 2020-10-16

**Source:** ISO/IEC JTC 1/SC 29/WG 03

**Expected action:** None

**Action due date:** None

**No. of pages:** 14 (with cover page)

**Email of Convenor:** young.L @ samsung . com

**Committee URL:** <https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg3>

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 03 MPEG SYSTEMS**

**ISO/IEC JTC 1/SC 29/WG 03 N** **0012**

**October 2020, Virtual**

|  |  |
| --- | --- |
| **Title** | **Advance signalling of MPEG containers content** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **19692** |

# Introduction

We believe there is an opportunity to bring together a few media signaling properties. This relates to the advanced signaling in MPEG, the DASH profile for CMAF content, the work on CTA DPCTF to add media capabilities, work in 3GPP on 5G media streaming.

Let us briefly explain:

1. We have signaling in ISO BMFF/CMAF (especially in the header)
2. We have the W3C APIs, in particular isTypeSupported and media capabilities APIs
3. We have manifest signaling in DASH and HLS
4. We also have elementary stream signaling.
5. We have codecs independent code points

These should all have the same meaning.

1. The ISO BMFF Header or CMAF Header contains some data such as codecs parameter, width/height, transfer characteristics and so on.
2. How to set this, is part of a CMAF profile. I.e., a CMAF media profile must provide this information (possibly using part 15, etc.)
3. Preferably, the connection to outside, such as transfer characteristics is handled through CICP
4. The manifests should copy this information into some parameters (e.g. @codecs, @width, @framerate, etc.)
5. If you get a manifest, you should be able to map the parameters directly to the media capability APIs from W3C

My proposal would be to write an overview document that explains all of this addressing two issues:

1. Collecting on what capabilities are defined where (CICP, W3C, etc.)
2. How the signaling is mapped to container formats, especially ISO BMFF and CMAF (by reference to the specs)
3. How the signaling is mapped to manifests (by reference to specs)
4. How you can use the manifest/container information to map it to capabilities
5. A recipe how you proceed if you want to add new information (where to go an  how to do)

In the Annex there is a first doc with some pointers.

# Responses

## Undisclosed Response 1

Hm, some random thoughts here.

The coding world has typically considered rendering “out of scope” and data about it “merely informative”, so nothing, for example, in an HEVC profile tells you what transfer function is being used, or even if the coding is in RGB or YCbCr or something more exotic (matrix).

I was one of the original authors of the “codecs” MIME parameter, and obviously I have overseen the file format. I now think that the “codecs” parameter was perhaps a mistake, as it allows you to go into depth in the codecs, and their profiles and levels, but not in other aspects like rendering, and obviously you can’t talk about combinations (audio and video), or systems aspects (how big are your media segments?).

So overall, I am more enthused by the profiles parameter, which reports on brands; as the brand indicates “this file conforms to the requirements of the specification identified by the brand” and that spec. can (and in the case of CMAF, does) go into considerable detail about combinations, rendering, and systems aspects.

This is complicated by the new “combination brands” (you can finally say that the reader needs to support BOTH brand X and brand Y in order to process the file). I think we introduced syntax for that in the profiles parameter.

Obviously, one could dig further into the file format; find, for example, track width and height fields, or the colour box in the sample entry (for the transfer function, matrix, and primaries), or… but the list is probably endless. Audio people would want to know what’s in the channel assignment box, for example. Maybe they need to know if DRC information is present; maybe we need to tell clients “don’t process this unless you support DRC”. I am not sure. It seems a rat’s nest.

## Undisclosed Response 2

thanks for putting this together! Is the main purpose to align the multiple “entry points” for capabilities (FF headers, manifest, bitstream) as a first step?

What causese some headache every once in a while is the question of how to deal with “variable” capabilities, e.g. width/height or framerate changes over time while headers allow only static values.

Either way, I’m fine with an offline call and/or talking about that during the MPEG meeting.

## Undisclosed Response 3: the Profile parameter

we should surely file an issue at <<https://github.com/w3c/media-capabilities/issues>>

"If the MIME type does not imply a codec, the string MUST 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.”

<<https://w3c.github.io/media-capabilities/>>

## Undisclosed Response 4:

why are parameters banned?

Discussion of profiles parameter is here:

<https://github.com/w3c/media-capabilities/issues/98>

Feedback on W3C specs is welcome from non-members, you are free to comment.

The proposal in this thread was to create a JavaScript library manage the mappings.

# Proposal

It is proposed that MPEG does something by not everything. We need to be clear on our position.

In more details:

1. Codec-independent code points: We should promote those and they should be used in eth capability signaling in order to make sure they are properly expressed
2. Brands or codec strings: We should have a decision. My preference is codec strings as brands are not really accepted and result in to many open issues.
3. Associated very good specs with a few code points.
4. We need to be clear if the code points/signaling express:
   1. Stream capabilities
   2. Receiver capabilities
5. Do everything for audio and video and possibly other media types

Let us start a TR and AHG on this.

# Contents

[2 Scope 1](#_Toc37883478)

[3 Media Capabilities in DPCTF (Will, Thomas) 1](#_Toc37883479)

[Annex A: Device Capability Discovery (Informative) 1](#_Toc37883480)

[A.1. General 1](#_Toc37883481)

[A.2. Capability Discovery Options (not about signaling) 2](#_Toc37883482)

[A.2.1. Media Profile 2](#_Toc37883483)

[A.2.2. CMAF Header 3](#_Toc37883484)

[A.2.3. MIME Subparameters 3](#_Toc37883485)

[A.2.4. Media Capabilities 4](#_Toc37883486)

[A.2.5. Device Capability – Persistent Item Solution 6](#_Toc37883487)

[A.2.6. User Agent String 6](#_Toc37883488)

[A.2.7. WAVE Playback Capabilities 7](#_Toc37883489)

[A.2.8. Rendering and Display Capabilities 7](#_Toc37883490)

[A.3. Recommendations for Capability Discovery APIs 7](#_Toc37883491)

[4 DASH Profile for CMAF Content 9](#_Toc37883492)

[**8.X.4** **Adaptation Set Constraints** 9](#_Toc37883493)

[5 3GPP TS 26.512 10](#_Toc37883494)

[5.2.7 Capability discovery 10](#_Toc37883495)

[6 3GPP TS 26.116 11](#_Toc37883496)

[5.1.3 Adaptation Set Constraints 11](#_Toc37883497)

# Scope

This document collects existing aware information on ad advance signaling and media capability APIs.

# Media Capabilities in DPCTF

Annex A: Device Capability Discovery (Informative)

* 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:

* A WAVE content signaling as defined in the content specification.
* 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.
* The signaling in the CMAF Header, specifically:
  + the compatibility brands in the ftyp box, and
  + The information of the sample description box in the CMAF Header.

Note that the information may be duplicated on different levels.

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.

Some options:

* HTMLMediaElement.canPlayType to determine if a mimetype/codec is supported.
* SourceBuffer can be created to handle the media type we are interested in, so we should be using MediaSource.isTypeSupported.
  1. Capability Discovery Options (not about signaling)

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.

* + 1. 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 profile="<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 does not support the configuration signaling and requires an additional capability mechanism on which configuration is preferably used.

* + 1. 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, future-proof, but the drawback of not being human readable and possibly requires transmitting more information than the other approaches.

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.

* + 1. 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 mostly used options today because it has the advantages of enabling a progressive, detailed, compact and almost 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 the communication with the device platform. The application queries the device of the media profile can be played back using:

* <mediatype>/mp4 mime-subparameters

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.

* + 1. Media Capabilities

The Media Capabilities API[[1]](#footnote-2) provides an improved alternative to the isSupportedType()API for determing 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.encodingInfo() method is called. The method takes an instance of a MediaDecodingConfiguration object 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 enviropment, the MediaDecodingType of the MediaDecodingConfiguration instance should be set to an enumerated string value of “media-source”.

The video and audio configurations are defined as below:

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 MUST 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. An example of a valid contentType would be

'video/mp4;codecs=avc1.640028'

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 in the following ways:

* VideoConfiguration
  + contentType – by parsing the stsd (sample description) box.
  + width – by parsing the codec specific box (such as avc1) inside the stsd (sample description) box.
  + height – by parsing the codec specific box (such as avc1) inside the stsd (sample description) box.
  + bitrate - ?????
  + framerate - dividing the timescale extracted from the mvhd box by the default\_sample\_duration from the tfhd box, gives the framerate in frames per second.
* AudioConfiguration
  + contentType – by parsing the stsd (sample description) box.
    1. Device Capability – Persistent Item Solution

Another approach is to standardized device capabilities in a known format for storage by the manufacturer and recall in some manner available to the application.  The preferred approach uses standardizes key-value pairs for relevant player characteristics that can be communicated to servers via JavaScript APIs or Objects.  One option is to use the HTTP User Agent String as described in [HBBTV] 7.3.2.4.

* + 1. User Agent String

The approach can provide something that can be deployed on devices where the new media capabilities API (or some future derivative) isn't supported. Typically, when integrating a browser on to a device, the HTTP User Agent is something that can be set by the device vendor / integrator without changing the code of the browser.

The user agent string would indicate which WAVE media profiles are supported in the HTTP UA. One example of how this could be done would be to include 3 bit-fields, one each for video, audio and captions/subtitles. Each would have one bit for each WAVE media profile with '1' indicating supported and '0' indicating not supported. HTML pages can read the HTTP User Agent string via the Navigator.userAgent property.

The strengths of this specific proposal (relative to others) are:

* Simplicity.
* Ease of integration for device makers/integrators due to not needing changes to the browser code.
* Something WAVE specific can also indicate support for WAVE device playback requirements and not "just" some support for the codec (which can be tested via isTypeSupported and/or canPlayType).

The weaknesses of this specific proposal (relative to others) are:

* Generic hostility to using the UA string.
* The ease with which it can be faked.
* Does not expose the same level of detail as the media capabilities API such as "smooth" and “powerEfficient”.
* More work would be needed to quantify how much added value it has over isTypeSupported / canPlayType in practice.

There are also of course issues common to other approaches:

* That they do not expose support for options within a WAVE media profile

concerns that exposing the supported media profiles can be used in combination with other information to identify / track the user.

* + 1. WAVE Playback Capabilities

Finally, there may be defined a dedicated capability code for the platform that matches against the full requirements in this specification. While such an approach may provide the most stringent interoperability, at the same time adding yet another option to the already complex world of capability signaling was dispensed during the development of this specification.

* + 1. Rendering and Display Capabilities

For the case when remotely connected displays are used, the display capabilities (or audio rendering system) needs to be discovered from the user agent. It is unclear if this is possible in general and what the limitations are. May be a question towards HTML 5 API.

Discussion on HDMI details are for further study.

* 1. Recommendations for Capability Discovery APIs

Based on the discussions in clause 6.2.2, it is recommended that new devices support one of the following two API structures:

* Media Profile capability API as defined in clause 6.2.2.1. However, this requires an additional signaling for identifying configuration options.
* CMAF header API as defined in clause 6.2.2.2. However, this requires an additional signaling for identifying configuration options.

In both cases, if there is a YES response to the capability, the device shall support the playback requirements defined in this specification for a specific media profile. If the device responds with a no, then it should indicate what capabilities are not supported.

However, devices may not implement either of these two capability APIs, the response of the device for such an API may be unknown and in particular existing devices may not support such an API at all. If a media profile wants to enable playback on devices that do not support one of the two APIs as recommended above, then it is recommended that the media profile documents other means for capability detection for such a media profile, in particular using one of the following:

* MIME Subparameters as documented in clause 6.2.2.3
* Media Capability approach as documented in clause 6.2.2.4

Each media profile should provide sufficient information on how to use APIs for capability discovery in order to ensure the playback of the media profile following the requirements in this specification. Specifically, suitable capability discovery for existing devices is recommended to be added.

# DASH Profile for CMAF Content

**8.X.4 Adaptation Set Constraints**

If this profile applies to an Adaptation Set, then the following holds for this Adaptation Set:

* The @contentType shall set to the hdlr type of the CMAF Master Header of the Switching Set, i.e. to vide -> video, soun -> audio, subt/text -> text
* The @mimeType shall be compatible to "<contentType>/mp4, profiles='cmfc'"
* The @codecs parameter shall be set to the sample entry of the CMAF Master Header
* If the Content is protected, then **ContentProtection** element shall be present and set appropriately:
  + A **ContentProtection** element with @schemeIdUri equal to "urn:mpeg:dash:mp4protection:2011" and @value equal to cenc or cbcs depending on the encryption scheme used. 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.
* If the @contentType is video, then the following should apply
  + @maxWidth is set to the width in the CMAF TrackHeaderBox of the CMAF Master Header
  + @maxHeight is set to the height in the CMAF TrackHeaderBox of the CMAF Master Header
* If the @contentType is audio, then no additional requirements apply.
* Each CMAF Track k=1,…K in the Switching Set shall be mapped to exactly one Representation as defined in clause 8.X.3.
* If the @bitstreamSwitching is set to true for this Adaptation Set, then the included CMAF Switching Set shall follow the "CMAF switching set single initialization constraints" of clause 7.3.4.2 in ISO/IEC 23000-19.
* Either the @segmentAlignment or @subsegmentAlignment shall be set.
* Event Message Streams may be signaled with **InbandEventStream** elements.

# 3GPP TS 26.512

### 5.2.7 Capability discovery

#### 5.2.7.1 General

A 5GMSd client is expected to support capability discovery such that 5GMS-aware applications can identify if a specific media profile is supported. In order to identify if a media profile is supported, the 5GMS client may provide an API as defined in TS 26.512 [10] in via the M7d interface, for which the client can be queried with a specific MIME type string, if the media profile is supported.

The MIME types follow RFC 6381 [11].

A 5GMSd client should support at least one of the following capability discovery mechanisms for media profiles:

- If isTypeSupported() for the media profile with argument <profile> results in a yes, then the respective media profile is supported with the requirements defined in a specific clause.

- If isTypeSupported() for the media profile with argument <codec> results in a yes, then the respective media profile is supported with the requirements defined in a specific clause.

- If a conforming CMAF header is provided for playback initialization and 5GMS Client does not throw an error response, then the respective media profile is supported with the requirements defined in a specific clause.

For each media profile mentioned in clause 5.2.5, the <profile> parameter and the <codec> parameter are provided in the following. These parameters should be used in the capability exchange.

#### 5.2.7.2 Video media profiles

For AVC-HD,

- the <profile> parameter is compatible to video/mp4 profiles='cmfc'

- the <codecs> parameter is compatible to either 'avc1.640028' or 'avc3.640028'

For AVC-FullHD,

- the <profile> parameter is compatible to video/mp4 profiles='cmfc'

- the <codecs> parameter is compatible to either 'avc1.640029' or 'avc3.640029'

For AVC-UHD,

- the <profile> parameter is compatible to video/mp4 profiles='cmfc'

- the <codecs> parameter is compatible to either 'avc1.640028' or 'avc3.640028'

For HEVC-HD,

- the <profile> parameter is compatible to video/mp4 profiles='cmfc'

- the <codecs> parameter is compatible to either 'hvc1.1.2.L93.B0' or 'hev1.1.2.L93.B0'

For HEVC-FullHD,

- the <profile> parameter is compatible to video/mp4 profiles='cmfc'

- the <codecs> parameter is compatible to either 'hvc1.2.4.L123.B0' or 'hev1.2.4.L123.B0'

For HEVC-UHD,

- the <profile> parameter is compatible to video/mp4 profiles='cmfc'

- the <codecs> parameter is compatible to either 'hvc1.2.4.L153.B0' or 'hev1.2.4.L153.B0'

# 3GPP TS 26.116

### 5.1.3 Adaptation Set Constraints

For a video Adaptation Set, the following constraints apply:

- The @codecs parameter shall be present on Adaptation Set level and shall signal the maximum required capability to decode any Representation in the Adaptation Set. The @codecs parameter should be signalled on the representation level if different from the one on Adaptation Set level.

- The @profiles parameter may be present to signal the constraints for the Adaptation Set.

- The attributes @maxWidth and @maxHeight shall be present. They are expected be used to signal the original source content format. This means that they may exceed the actual largest size of any coded Representation in one Adaptation Set. More details for each Operation Point are provided.

- The @width and @height shall be signalled for each Representation (possibly defaulted on Adaptation Set level) and shall match the values of the maximum width and height in the Sample Description box of the contained Representation.

- The Chroma Format may be signalled. If signalled:

- An Essential or Supplemental Descriptor shall be used to signal the value by setting the @schemeIdURI attribute to urn:mpeg:mpegB:cicp:MatrixCoefficients as defined ISO/IEC 23001-8 [10] and the @value attribute according to Table 4 of ISO/IEC 23001-8 [10]. The values shall match the values set in the VUI.

- The signalling shall be on Adaptation Set level.

- The Color Primaries and Transfer Function shall be signalled unless ITU-R BT.709 is used. If signalled:

- An Essential or Supplemental Descriptor shall be used to signal the value by setting the @schemeIdURI attribute to urn:mpeg:mpegB:cicp:ColourPrimaries and urn:mpeg:mpegB:cicp:TransferCharacteristics as defined ISO/IEC 23001-8 [10] and the @value attribute according to Table 4 of ISO/IEC 23001-8 [10]. The values shall match the values set in the VUI.

- The signalling shall be on Adaptation Set level only, i.e. the value shall not be different for different Representations in one Adaptation Set.

- The maximum frame rate may be signalled on Adaptation Set using the @maxFrameRate attribute.

- The @frameRate shall be signalled for each Representation (possibly defaulted on Adaptation Set level). In one Adaptation Set, only frame rates shall be present from one of the following subsets:

- 24 Hz with proposed signalling @frameRate="24"

- 25 Hz, 50 Hz with proposed signalling @frameRate="25" or @frameRate="50",

- 30 Hz, 60 Hz with proposed signalling @frameRate="30" or @frameRate="60",

- 24/1.001 Hz with proposed signalling @frameRate="24000/1001",

- 30/1.001 Hz, 60/1.001 Hz with proposed signalling @frameRate="30000/1001" or @frameRate="60000/1001".

- Random Access Points shall be signalled by @startsWithSAP set to 1, 2 or 3.

1. <https://wicg.github.io/media-capabilities/> [↑](#footnote-ref-2)