**3GPP TSG SA WG4#130 S4-241896**

**online, Teams, 19 - 23 August 2024**

**Agenda item:** 9.5

**Source:** Qualcomm Incorporated

**Title:** [VOPS] 3GPP Video Codecs as Web Codecs

**Document for** Agreement

# Introduction

During SA#103 the new work item on “Video Operating Points - Harmonization and Stereo MV-HEVC” was approved in SP-240060. The objectives of this work are to:

1. *Harmonize and include as needed all the SA4 video operating points, such as Video profiles Operation Points, Video Operation Points, video encode and decode capabilities etc., which are currently scattered in various SA4 specifications (e.g. TS 26.116, TS 26.118, TS 26.119, TS 26.143, and TS 26.511), into a new specification that will be home to all such video operating points and upgrade HEVC-based levels based on industry practices.*
2. Define the MV-HEVC capability in this new specification.
3. Then add and harmonize stereoscopic MV-HEVC (potentially with auxiliary information, e.g. alpha channels) encode/decode operating points, capabilities, streaming (e.g. CMAF, DASH) and transport aspects for:
	1. 5G-media streaming profiles, codecs, and formats (TS 26.511)
	2. Media capabilities for AR devices (TS 26.119)
	3. Video messaging media profiles (TS 26.143)
4. Perform the above work in coordination with related SDOs and industrial fora such as MPEG, DASH-IF, CTA-WAVE, and IETF, and by referencing the related specifications, e.g. the Common Media Application Format (CMAF) and the ISO base media file format (ISOBMFF), among others.

During SA4#129-e, document S4-241479 was agreed with the following proposal:

1. To support registration of relevant 3GPP video codecs (including AVC, HEVC, MV-HEVC) together with MPEG as web codecs including
	1. Definition of codecs strings
	2. Definition of video chunks
	3. Definition of video decoder config
	4. Definition of extensions of the video encoder config
	5. Definition of extensions of the video decoder config
2. Consider to document codecs with MSE principles and support testing of codecs.
3. In VOPS, an informative Annex is created.
4. However, as pull requests to the registry are needed, it is proposed to create a registry as part of 3GPP/ETSI githubs.
5. The registration process of codecs should be initiated officially by 3GPP through an LS at appropriate time.

This document follows-up on the above proposal. Some remaining content from S4-241479 is kept as reference.

This document addresses some comments during the first session:

|  |  |  |  |
| --- | --- | --- | --- |
| **[S4-241896](https://www.3gpp.org/ftp/TSG_SA/WG4_CODEC/TSGS4_130_Orlando/Docs/S4-241896.zip)** | [VOPS] 3GPP Video Codecs as Web Codecs | Qualcomm Technologies Int | Thomas Stockhammer |

 **Revisions**: none

**Online Discussion**:

* **Session 2: 16:00-18:00**
* Thomas presents.
* Stefan (Döhla): Is it on top of MPEG definitions?
* Thomas: W3C WebCodes information is a bit superficial. This could be much more detailed with MPEG and JVET information, e.g. the chunk format.
* Krasimir: Is the proposal to add something to CMAF?
* Thomas: No, I would like to encourage the clarification of the HEVC extension of the WebCodecs based on the work in CMAF.
* Krasimir: How do you see the collaboration with CTA WAVE?
* Thomas: We need test sequences and possibly hosted by CTA WAVE. We need to engage with them.
* Dimitri: Was there some communication between W3C and MPEG/JVET?
* Thomas: Not to my knowledge. MPEG/JVET should have a look.
* Dimitri: Is it also going to the file format:
* Thomas: No, just bitstream level.
* Emmanuel: WebCodecs is bitstream level, no dependency on file format. For part 15, they use the sample format.
* Dimitri: Ok, but then the parsing of the length depends on some parameter in the decoder configuration.
* Thomas: Let’s come on this on Thursday.

**Decision**:

* Session 2: parked.

**[S4-241896](https://www.3gpp.org/ftp/TSG_SA/WG4_CODEC/TSGS4_130_Orlando/Docs/S4-241896.zip)** is **noted/agreed/revised**.

# What are Web Codecs?

## 2.1 Introduction

The WebCodecs API (https://www.w3.org/TR/webcodecs/) is a W3C working draft that is currently under development. It specifies a powerful web API that provides developers with low-level access to the individual frames of a video stream and chunks of audio. It is particularly useful for web applications that require full control over the way media is processed, such as video or audio editors, and video conferencing applications.

The WebCodecs API provides access to codecs that are already in the browser, eliminating the need for additional software codecs and leveraging the existing hardware acceleration on the device. It gives access to raw video frames, chunks of audio data, image decoders, audio and video encoders, and decoders.

The WebCodecs API uses an asynchronous processing model. Each instance of an encoder or decoder maintains an internal, independent processing queue. Methods named configure(), encode(), decode(), and flush() operate asynchronously by appending control messages to the end of the queue, while methods named reset() and close() synchronously abort all pending work and purge the processing queue.

The WebCodecs API provides several video related interfaces:

* VideoDecoder: Decodes EncodedVideoChunk objects.
* VideoEncoder: Encodes VideoFrame objects.
* EncodedVideoChunk: Represents codec-specific encoded video bytes.
* VideoFrame: Represents a frame of unencoded video data.
* VideoColorSpace: Represents the color space of a video frame.

The following table provides a simple example code for the usage of WebCodecs to demonstrate the functionality of the WebCodecs API:

|  |
| --- |
| // Create a new VideoDecoder and configure itconst init = { output: handleFrame, error: (e) => { console.log(e.message); },};const config = { codec: "hevc", codedWidth: 1280, codedHeight: 720};let decoder = new VideoDecoder(init);decoder.configure(config);// Create a new VideoEncoder and configure itlet encoder = new VideoEncoder({ output: (chunk) => { const buffer = new ArrayBuffer(chunk.byteLength); chunk.copyTo(buffer); chunks.push(buffer); }, error: (e) => console.error(e.message)});encoder.configure({ codec: 'hevc', width: 1280, height: 720, bitrate: 2000000, framerate: 25});// Encode every image as a frame track.requestFrame().then((frame) => { encoder.encode(frame, {keyFrame: true}); frame.close(); }); // Create a video from it encoder.flush().then(() => { const blob = new Blob(chunks, {type: 'video/webm; codecs=vp8'}); const url = URL.createObjectURL(blob); decoder.decode(new EncodedVideoChunk({ type: 'key', timestamp: 0, data: blob })); });}).catch((error) => { console.error("Error: ", error);}); |

A full example can be found under <https://bouazizi.dev/webcodecs/>

# Video Related Interfaces

## Video Decoder Interface

The [video decoder interface](https://www.w3.org/TR/webcodecs/#videodecoder-interface) is defined as an API that allows to apply three main methods

* Configure: Enqueues a control message to configure the video decoder for decoding chunks as described by config.
* Decode: Enqueues a control message to decode the given chunk.
* IsConfigSupported: Returns a promise indicating whether the provided config is supported by the User Agent.

[[Exposed](https://webidl.spec.whatwg.org/#Exposed)=(Window,DedicatedWorker), [SecureContext](https://webidl.spec.whatwg.org/#SecureContext)]

interface ***VideoDecoder*** : [EventTarget](https://dom.spec.whatwg.org/#eventtarget) {

 [constructor](https://www.w3.org/TR/webcodecs/#dom-videodecoder-videodecoder)([VideoDecoderInit](https://www.w3.org/TR/webcodecs/#dictdef-videodecoderinit) ***init***);

 readonly attribute [CodecState](https://www.w3.org/TR/webcodecs/#enumdef-codecstate) [state](https://www.w3.org/TR/webcodecs/#dom-videodecoder-state);

 readonly attribute [unsigned long](https://webidl.spec.whatwg.org/#idl-unsigned-long) [decodeQueueSize](https://www.w3.org/TR/webcodecs/#dom-videodecoder-decodequeuesize);

 attribute [EventHandler](https://html.spec.whatwg.org/multipage/webappapis.html#eventhandler) [ondequeue](https://www.w3.org/TR/webcodecs/#dom-videodecoder-ondequeue);

 [undefined](https://webidl.spec.whatwg.org/#idl-undefined) [configure](https://www.w3.org/TR/webcodecs/#dom-videodecoder-configure)([VideoDecoderConfig](https://www.w3.org/TR/webcodecs/#dictdef-videodecoderconfig) ***config***);

 [undefined](https://webidl.spec.whatwg.org/#idl-undefined) [decode](https://www.w3.org/TR/webcodecs/#dom-videodecoder-decode)([EncodedVideoChunk](https://www.w3.org/TR/webcodecs/#encodedvideochunk) ***chunk***);

 [Promise](https://webidl.spec.whatwg.org/#idl-promise)<[undefined](https://webidl.spec.whatwg.org/#idl-undefined)> [flush](https://www.w3.org/TR/webcodecs/#dom-videodecoder-flush)();

 [undefined](https://webidl.spec.whatwg.org/#idl-undefined) [reset](https://www.w3.org/TR/webcodecs/#dom-videodecoder-reset)();

 [undefined](https://webidl.spec.whatwg.org/#idl-undefined) [close](https://www.w3.org/TR/webcodecs/#dom-videodecoder-close)();

 static [Promise](https://webidl.spec.whatwg.org/#idl-promise)<[VideoDecoderSupport](https://www.w3.org/TR/webcodecs/#dictdef-videodecodersupport)> [isConfigSupported](https://www.w3.org/TR/webcodecs/#dom-videodecoder-isconfigsupported)([VideoDecoderConfig](https://www.w3.org/TR/webcodecs/#dictdef-videodecoderconfig) ***config***);

};

dictionary ***VideoDecoderInit*** {

 required [VideoFrameOutputCallback](https://www.w3.org/TR/webcodecs/#callbackdef-videoframeoutputcallback) ***output***;

 required [WebCodecsErrorCallback](https://www.w3.org/TR/webcodecs/#callbackdef-webcodecserrorcallback) ***error***;

};

callback ***VideoFrameOutputCallback*** = [undefined](https://webidl.spec.whatwg.org/#idl-undefined)([VideoFrame](https://www.w3.org/TR/webcodecs/#videoframe) ***output***);

The configuration of the codec is here

dictionary ***VideoDecoderConfig*** {

 required [DOMString](https://webidl.spec.whatwg.org/#idl-DOMString) [codec](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-codec);

 [AllowSharedBufferSource](https://webidl.spec.whatwg.org/#AllowSharedBufferSource) [description](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-description);

 [[EnforceRange](https://webidl.spec.whatwg.org/#EnforceRange)] [unsigned long](https://webidl.spec.whatwg.org/#idl-unsigned-long) [codedWidth](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-codedwidth);

 [[EnforceRange](https://webidl.spec.whatwg.org/#EnforceRange)] [unsigned long](https://webidl.spec.whatwg.org/#idl-unsigned-long) [codedHeight](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-codedheight);

 [[EnforceRange](https://webidl.spec.whatwg.org/#EnforceRange)] [unsigned long](https://webidl.spec.whatwg.org/#idl-unsigned-long) [displayAspectWidth](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-displayaspectwidth);

 [[EnforceRange](https://webidl.spec.whatwg.org/#EnforceRange)] [unsigned long](https://webidl.spec.whatwg.org/#idl-unsigned-long) [displayAspectHeight](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-displayaspectheight);

 [VideoColorSpaceInit](https://www.w3.org/TR/webcodecs/#dictdef-videocolorspaceinit) [colorSpace](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-colorspace);

 [HardwareAcceleration](https://www.w3.org/TR/webcodecs/#enumdef-hardwareacceleration) [hardwareAcceleration](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-hardwareacceleration) = "no-preference";

 [boolean](https://webidl.spec.whatwg.org/#idl-boolean) [optimizeForLatency](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-optimizeforlatency);

};

A codec string describes a given codec format to be used for encoding or decoding.

A **valid codec string** must meet the following conditions.

1. Is valid per the relevant codec specification (see examples below).
2. It describes a single codec.
3. It is unambiguous about codec profile, level, and constraint bits for codecs that define these concepts.

NOTE: In other media specifications, codec strings historically accompanied a [MIME type](https://mimesniff.spec.whatwg.org/#mime-type) as the "codecs=" parameter ([isTypeSupported()](https://www.w3.org/TR/media-source-2/#dom-mediasource-istypesupported), [canPlayType()](https://html.spec.whatwg.org/multipage/media.html#dom-navigator-canplaytype)) [[RFC6381]](https://www.w3.org/TR/webcodecs/#biblio-rfc6381). In this specification, encoded media is not containerized; hence, only the value of the codecs parameter is accepted.

NOTE: Encoders for codecs that define level and constraint bits have flexibility around these parameters, but won’t produce bitstreams that have a higher level or are less constrained than requested.

The format and semantics for codec strings are defined by codec registrations listed in the [[WEBCODECS-CODEC-REGISTRY]](https://www.w3.org/TR/webcodecs/#biblio-webcodecs-codec-registry). A compliant implementation may support any combination of codec registrations or none at all.

## Video Encoder Interface

Similar as for the decoder, [an API for the encoder](https://www.w3.org/TR/webcodecs/#videoencoder-interface) is defined

[[Exposed](https://webidl.spec.whatwg.org/#Exposed)=(Window,DedicatedWorker), [SecureContext](https://webidl.spec.whatwg.org/#SecureContext)]

interface ***VideoEncoder*** : [EventTarget](https://dom.spec.whatwg.org/#eventtarget) {

 [constructor](https://www.w3.org/TR/webcodecs/#dom-videoencoder-videoencoder)([VideoEncoderInit](https://www.w3.org/TR/webcodecs/#dictdef-videoencoderinit) ***init***);

 readonly attribute [CodecState](https://www.w3.org/TR/webcodecs/#enumdef-codecstate) [state](https://www.w3.org/TR/webcodecs/#dom-videoencoder-state);

 readonly attribute [unsigned long](https://webidl.spec.whatwg.org/#idl-unsigned-long) [encodeQueueSize](https://www.w3.org/TR/webcodecs/#dom-videoencoder-encodequeuesize);

 attribute [EventHandler](https://html.spec.whatwg.org/multipage/webappapis.html#eventhandler) [ondequeue](https://www.w3.org/TR/webcodecs/#dom-videoencoder-ondequeue);

 [undefined](https://webidl.spec.whatwg.org/#idl-undefined) [configure](https://www.w3.org/TR/webcodecs/#dom-videoencoder-configure)([VideoEncoderConfig](https://www.w3.org/TR/webcodecs/#dictdef-videoencoderconfig) ***config***);

 [undefined](https://webidl.spec.whatwg.org/#idl-undefined) [encode](https://www.w3.org/TR/webcodecs/#dom-videoencoder-encode)([VideoFrame](https://www.w3.org/TR/webcodecs/#videoframe) ***frame***, optional [VideoEncoderEncodeOptions](https://www.w3.org/TR/webcodecs/#dictdef-videoencoderencodeoptions) ***options*** = {});

 [Promise](https://webidl.spec.whatwg.org/#idl-promise)<[undefined](https://webidl.spec.whatwg.org/#idl-undefined)> [flush](https://www.w3.org/TR/webcodecs/#dom-videoencoder-flush)();

 [undefined](https://webidl.spec.whatwg.org/#idl-undefined) [reset](https://www.w3.org/TR/webcodecs/#dom-videoencoder-reset)();

 [undefined](https://webidl.spec.whatwg.org/#idl-undefined) [close](https://www.w3.org/TR/webcodecs/#dom-videoencoder-close)();

 static [Promise](https://webidl.spec.whatwg.org/#idl-promise)<[VideoEncoderSupport](https://www.w3.org/TR/webcodecs/#dictdef-videoencodersupport)> [isConfigSupported](https://www.w3.org/TR/webcodecs/#dom-videoencoder-isconfigsupported)([VideoEncoderConfig](https://www.w3.org/TR/webcodecs/#dictdef-videoencoderconfig) ***config***);

};

dictionary ***VideoEncoderInit*** {

 required [EncodedVideoChunkOutputCallback](https://www.w3.org/TR/webcodecs/#callbackdef-encodedvideochunkoutputcallback) ***output***;

 required [WebCodecsErrorCallback](https://www.w3.org/TR/webcodecs/#callbackdef-webcodecserrorcallback) ***error***;

};

callback ***EncodedVideoChunkOutputCallback*** =

 [undefined](https://webidl.spec.whatwg.org/#idl-undefined) ([EncodedVideoChunk](https://www.w3.org/TR/webcodecs/#encodedvideochunk) ***chunk***,

 optional [EncodedVideoChunkMetadata](https://www.w3.org/TR/webcodecs/#dictdef-encodedvideochunkmetadata) ***metadata*** = {});

The configuration for the encoder is provided here

dictionary ***VideoEncoderConfig*** {

 required [DOMString](https://webidl.spec.whatwg.org/#idl-DOMString) [codec](https://www.w3.org/TR/webcodecs/#dom-videoencoderconfig-codec);

 [[EnforceRange](https://webidl.spec.whatwg.org/#EnforceRange)] required [unsigned long](https://webidl.spec.whatwg.org/#idl-unsigned-long) [width](https://www.w3.org/TR/webcodecs/#dom-videoencoderconfig-width);

 [[EnforceRange](https://webidl.spec.whatwg.org/#EnforceRange)] required [unsigned long](https://webidl.spec.whatwg.org/#idl-unsigned-long) [height](https://www.w3.org/TR/webcodecs/#dom-videoencoderconfig-height);

 [[EnforceRange](https://webidl.spec.whatwg.org/#EnforceRange)] [unsigned long](https://webidl.spec.whatwg.org/#idl-unsigned-long) [displayWidth](https://www.w3.org/TR/webcodecs/#dom-videoencoderconfig-displaywidth);

 [[EnforceRange](https://webidl.spec.whatwg.org/#EnforceRange)] [unsigned long](https://webidl.spec.whatwg.org/#idl-unsigned-long) [displayHeight](https://www.w3.org/TR/webcodecs/#dom-videoencoderconfig-displayheight);

 [[EnforceRange](https://webidl.spec.whatwg.org/#EnforceRange)] [unsigned long long](https://webidl.spec.whatwg.org/#idl-unsigned-long-long) [bitrate](https://www.w3.org/TR/webcodecs/#dom-videoencoderconfig-bitrate);

 [double](https://webidl.spec.whatwg.org/#idl-double) [framerate](https://www.w3.org/TR/webcodecs/#dom-videoencoderconfig-framerate);

 [HardwareAcceleration](https://www.w3.org/TR/webcodecs/#enumdef-hardwareacceleration) [hardwareAcceleration](https://www.w3.org/TR/webcodecs/#dom-videoencoderconfig-hardwareacceleration) = "no-preference";

 [AlphaOption](https://www.w3.org/TR/webcodecs/#enumdef-alphaoption) [alpha](https://www.w3.org/TR/webcodecs/#dom-videoencoderconfig-alpha) = "discard";

 [DOMString](https://webidl.spec.whatwg.org/#idl-DOMString) [scalabilityMode](https://www.w3.org/TR/webcodecs/#dom-videoencoderconfig-scalabilitymode);

 [VideoEncoderBitrateMode](https://www.w3.org/TR/webcodecs/#enumdef-videoencoderbitratemode) [bitrateMode](https://www.w3.org/TR/webcodecs/#dom-videoencoderconfig-bitratemode) = "variable";

 [LatencyMode](https://www.w3.org/TR/webcodecs/#enumdef-latencymode) [latencyMode](https://www.w3.org/TR/webcodecs/#dom-videoencoderconfig-latencymode) = "quality";

 [DOMString](https://webidl.spec.whatwg.org/#idl-DOMString) [contentHint](https://www.w3.org/TR/webcodecs/#dom-videoencoderconfig-contenthint);

};

# Codec Registration Procedure

The codec registration procedure for new codecs is defined by W3C in <https://www.w3.org/TR/webcodecs-codec-registry/>.

The [registration requirements](https://www.w3.org/TR/webcodecs-codec-registry/#registration-entry-requirements) are request:

* Codec String and specification
* The codec string has certain requirements
* Each registration must include
	+ Recognized codec strings
	+ EncodedVideoChunk internal data
	+ VideoDecoderConfig description bytes
	+ Expectations for EncodedVideoChunk
* Registration may include description of extensions to VideoEncoderConfig dictionaries
* Candidate entries must be announced by filing an issue in the [WebCodecs GitHub issue tracker](https://github.com/w3c/webcodecs/issues/) so they can be discussed and evaluated for compliance before being added to the registry.

The [video codec registry](https://www.w3.org/TR/webcodecs-codec-registry/#video-codec-registry) includes AV1, H.264/AVC, H.265/HEVC, VP8, VP9

As an example HEVC codec registrations, please go here: <https://www.w3.org/TR/webcodecs-hevc-codec-registration/>

# MSE related discussion

## Relation to VOPS and FS\_ACAPI work

3GPP codecs can be viewed as media service enablers.

Key aspects are the ability to

1. Use the encoder and decoder independently of a specific service
2. Allow to use and configure the encoder and decoder through APIs
3. Consider the ability to create unit tests

These aspects should be taken into account when defining codecs, decoding and encoding capabilities.

## Interface and Conformance View

A first view of interfaces and conformance is provided in the below diagram.



# Proposal

To follow-up on the agreed proposal, the following aspects are proposed from SA4#130

For the different operation points, the following information is collected

1. To support registration of relevant 3GPP video codecs (including AVC, HEVC, MV-HEVC) together with MPEG as web codecs including
	1. Definition of codecs strings
	2. Definition of video chunks
	3. Definition of video decoder config
	4. Definition of extensions of the video encoder config
	5. Definition of extensions of the video decoder config

It is proposed to use the table as a starting point to be added to Annex B. A pCR will be provided for an upcoming meeting.

|  |  |  |  |
| --- | --- | --- | --- |
| Operating Point | Codecs String | Video Chunk | Video Decoder Config |
| 3GPP-AVC-HDTV-CMAF | 'avc1.640029' or 'avc3.640029' | CMAF Fragment or CMAF Random Access chunk for AVC | CMAF Header for AVC |
| 3GPP-HEVC-HDTV-CMAF | 'hvc1.2.4.L123.B0' or 'hev1.2.4.L123.B0' | CMAF Fragment or CMAF Random Access chunk for HEVC | CMAF Header for HEVC |
| 3GPP-HEVC-HD-HDR-CMAF | 'hvc1.2.4.L123.B0' or 'hev1.2.4.L123.B0' | CMAF Fragment or CMAF Random Access chunk for HEVC | CMAF Header for HEVC |
| 3GPP-HEVC-UHD-HDR-CMAF | 'hvc1.2.4.L153.B0' or 'hev1.2.4.L153.B0' | CMAF Fragment or CMAF Random Access chunk for HEVC | CMAF Header for HEVC |
| 3GPP-HEVC-3DTV-CMAF | tbd | Tbd | Tbd |
| 3GPP-MVHEVC-3DTV-CMAF | Tbd | tbd | tbd |

Encoder configuration and extended decoder configuration is for further study.

1. Consider to document codecs with MSE principles and support testing of codecs.

Alignment with CTA WAVE playback capabilities for CMAF Test content and playback.

It is proposed to rely on CTA WAVE to develop CMAF Test Vectors for the VOPS profile, and to identify whether existing test material is available that conforms to the operation points. A follow-up analysis will be done in an upcoming meeting. Also for playback, this should be coordinate with CTA WAVE.

1. In VOPS, an informative Annex is created.

Add the above informationto Annex.

It is proposed to address this in a pCR in an upcoming meeting.

1. However, as pull requests to the registry are needed, it is proposed to create a registry as part of 3GPP/ETSI githubs.

No actions proposed, as needs to be coordinated with MPEG/JVET.

It is proposed that MPEG experts address this issue at an upcoming MPEG meeting.

1. The registration process of codecs should be initiated officially by 3GPP through an LS at appropriate time.

This can happen at a later time.