ISO/IEC JTC 1/SC 29/WG 03 N0514

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

**Document type:** Output Document

**Title:** Technologies under consideration for VDI

**Status:** Approved

**Date of document:** 2022-01-21

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

**No. of pages:** 21 (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 N0514**

**January 20222, Virtual**

|  |  |
| --- | --- |
| **Title** | **Technologies under consideration for VDI** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **21330** |

# Introduction

This document collects technologies under consideration for the development of ISO/IEC 23090-13, a.k.a. MPEG-I part 13: Video Decoding Interfaces for Immersive Media.

Contents

[1 Introduction 1](#_Toc93681665)

[2 Relevant W3C Recommendation and their relationship with VDI (from m56783) 1](#_Toc93681666)

[3 Khronos Vulkan Video extension (from m57544) 7](#_Toc93681667)

[4 VDI System Decoder Model (m58863 from MPEG#137) 10](#_Toc93681668)

[5 Layer composite position info SEI message (from JVET-S0107) 13](#_Toc93681670)

[6 Composite picture information SEI message (from JVET-T0049) 17](#_Toc93681671)

# Relevant W3C Recommendation and their relationship with VDI (from m56783)

## Media Source Extensions [1]

This specification extends [HTMLMediaElement](https://www.w3.org/TR/html51/semantics-embedded-content.html#htmlmediaelement-htmlmediaelement) [[*HTML51*](https://www.w3.org/TR/media-source/#bib-HTML51)] to allow JavaScript to generate media streams for playback. Allowing JavaScript to generate streams facilitates a variety of use cases like adaptive streaming and time shifting live streams.

### Relevance to VDI

MSE is relevant because it is meant to facilitate the manipulating the HTML media element in a buffer level - to facilitate advanced playback and buffering functionalities.

Most functionalities of MSE are around the **MediaSource** object that is acting as a source for the HTMLMediaElement. By itself it does not have any way to interpret the content, as such it requires a SourceBuffer object to be attached to it. Therefore, most of the methods and events of the MediaSource are designed with the purpose of “interfacing” the attached SourceBuffer objects with the HTML media element.

For the sake of completeness, the events defined for MediaSource are attached bellow:

|  |  |  |
| --- | --- | --- |
| **Event name** | **Interface** | **Dispatched when...** |
| *sourceopen* | Event | [readyState](https://www.w3.org/TR/media-source/#dom-readystate) transitions from ["closed"](https://www.w3.org/TR/media-source/#idl-def-ReadyState.closed) to ["open"](https://www.w3.org/TR/media-source/#idl-def-ReadyState.open) or from ["ended"](https://www.w3.org/TR/media-source/#idl-def-ReadyState.ended) to ["open"](https://www.w3.org/TR/media-source/#idl-def-ReadyState.open). |
| *sourceended* | Event | [readyState](https://www.w3.org/TR/media-source/#dom-readystate) transitions from ["open"](https://www.w3.org/TR/media-source/#idl-def-ReadyState.open) to ["ended"](https://www.w3.org/TR/media-source/#idl-def-ReadyState.ended). |
| *sourceclose* | Event | [readyState](https://www.w3.org/TR/media-source/#dom-readystate) transitions from ["open"](https://www.w3.org/TR/media-source/#idl-def-ReadyState.open) to ["closed"](https://www.w3.org/TR/media-source/#idl-def-ReadyState.closed) or ["ended"](https://www.w3.org/TR/media-source/#idl-def-ReadyState.ended) to ["closed"](https://www.w3.org/TR/media-source/#idl-def-ReadyState.closed). |

Since the **SourceBuffer** object is the one handling the streams prior going to the MediaSource, it is more relevant to the scope of VDI. **SourceBuffe**r takes as inputs **media segments** – which in the scope of this specification consist of a series of coded frames with a duration assigned. Most functionality of a Source Buffer is achieved by updating the objects with new media segments, and changing the way they are updated (if needed) to allow non-timestamp-based representations of frames. To provide a better overview of the **SourceBuffer**, following is the interface as defined in the specification:

interface SourceBuffer : EventTarget {

attribute [AppendMode](https://www.w3.org/TR/media-source/#idl-def-appendmode) [mode](https://www.w3.org/TR/media-source/#dom-sourcebuffer-mode);

readonly attribute [boolean](https://www.w3.org/TR/WebIDL-1/#idl-boolean) [updating](https://www.w3.org/TR/media-source/#dom-sourcebuffer-updating);

readonly attribute [TimeRanges](https://www.w3.org/TR/html51/semantics-embedded-content.html#timeranges-timeranges) [buffered](https://www.w3.org/TR/media-source/#dom-sourcebuffer-buffered);

attribute [double](https://www.w3.org/TR/WebIDL-1/#idl-double) [timestampOffset](https://www.w3.org/TR/media-source/#dom-sourcebuffer-timestampoffset);

readonly attribute [AudioTrackList](https://www.w3.org/TR/html51/semantics-embedded-content.html#audiotracklist-audiotracklist) [audioTracks](https://www.w3.org/TR/media-source/#dom-sourcebuffer-audiotracks);

readonly attribute [VideoTrackList](https://www.w3.org/TR/html51/semantics-embedded-content.html#videotracklist-videotracklist) [videoTracks](https://www.w3.org/TR/media-source/#dom-sourcebuffer-videotracks);

readonly attribute [TextTrackList](https://www.w3.org/TR/html51/semantics-embedded-content.html#texttracklist-texttracklist) [textTracks](https://www.w3.org/TR/media-source/#dom-sourcebuffer-texttracks);

attribute [double](https://www.w3.org/TR/WebIDL-1/#idl-double) [appendWindowStart](https://www.w3.org/TR/media-source/#dom-sourcebuffer-appendwindowstart);

attribute [unrestricted double](https://www.w3.org/TR/WebIDL-1/#idl-unrestricted-double) [appendWindowEnd](https://www.w3.org/TR/media-source/#dom-sourcebuffer-appendwindowend);

attribute [EventHandler](https://www.w3.org/TR/html51/webappapis.html#events-event-handlers) [onupdatestart](https://www.w3.org/TR/media-source/#dom-sourcebuffer-onupdatestart);

attribute [EventHandler](https://www.w3.org/TR/html51/webappapis.html#events-event-handlers) [onupdate](https://www.w3.org/TR/media-source/#dom-sourcebuffer-onupdate);

attribute [EventHandler](https://www.w3.org/TR/html51/webappapis.html#events-event-handlers) [onupdateend](https://www.w3.org/TR/media-source/#dom-sourcebuffer-onupdateend);

attribute [EventHandler](https://www.w3.org/TR/html51/webappapis.html#events-event-handlers) [onerror](https://www.w3.org/TR/media-source/#dom-sourcebuffer-onerror);

attribute [EventHandler](https://www.w3.org/TR/html51/webappapis.html#events-event-handlers) [onabort](https://www.w3.org/TR/media-source/#dom-sourcebuffer-onabort);

[void](https://www.w3.org/TR/WebIDL-1/#idl-void) [appendBuffer](https://www.w3.org/TR/media-source/#dom-sourcebuffer-appendbuffer)([BufferSource](https://www.w3.org/TR/WebIDL-1/" \l "common-BufferSource) data);

[void](https://www.w3.org/TR/WebIDL-1/#idl-void) [abort](https://www.w3.org/TR/media-source/#dom-sourcebuffer-abort)();

[void](https://www.w3.org/TR/WebIDL-1/#idl-void) [remove](https://www.w3.org/TR/media-source/#dom-sourcebuffer-remove)([double](https://www.w3.org/TR/WebIDL-1/#idl-double) start, [unrestricted double](https://www.w3.org/TR/WebIDL-1/#idl-unrestricted-double) end);

};

### Conclusion

It is relevant keep monitoring the MSE for future developments and when VDI matures to contact W3C proponents for possible technical alignment of features.

## Metadata API for Media Resources [2]

### Abstract

This specification defines an API to access metadata information related to media resources on the Web. The overall purpose is to provide developers with a convenient access to metadata information stored in different metadata formats. The API provides means to access the set of metadata properties defined in the [Ontology for Media Resources 1.0](http://www.w3.org/TR/mediaont-10/) specification. These [properties](http://www.w3.org/TR/mediaont-10/#core-property-definitions) are used as a pivot vocabulary in this API. The core of this specification is the definition of API interfaces for retrieving metadata information in synchronous and asynchronous modes. It also defines interfaces for structured return types along with the specification of the behavior of an API implementation.

### Relevance to VDI

Even though this specification precedes MSE in conception it was recently revived with promises for implementations and testing by browser vendors. The main goal of this API is to interface media metadata (mostly with semantical properties), however, there is consideration for technical properties, with current examples including values for FrameRate, AverageBitRate, NumTracks etc. As such it might be useful in the future to accommodate technical properties that would enable/optimize adoption of VDI (e.g. recommended number of decoders).

### Conclusion

For now, Metadata API is not interesting for VDI, but in the (unlikely?) case that it gets traction in the future we might want to use it for VDI signaling.

## Media Capabilities [3]

This specification intends to provide APIs to allow websites to make an optimal decision when picking media content for the user. The APIs will expose information about the decoding and encoding capabilities for a given format but also output capabilities to find the best match based on the device’s display.

### Relevance to VDI

This is an active (and new) specification that is probably the most relevant to VDI – alongside MSE. It seems to be overlapping with the Metadata API in that it can signal supported/available bitrates, codecs etc. with a focus on decoder and encoder capabilities. The high level of the interface is defined as following:

### Media Capabilities Interface

[[Exposed](https://heycam.github.io/webidl/#Exposed)=(Window, Worker)]

interface *MediaCapabilities* {

[[NewObject](https://heycam.github.io/webidl/" \l "NewObject)] [Promise](https://heycam.github.io/webidl/#idl-promise)<[MediaCapabilitiesDecodingInfo](https://www.w3.org/TR/media-capabilities/" \l "dictdef-mediacapabilitiesdecodinginfo)> *decodingInfo*([MediaDecodingConfiguration](https://www.w3.org/TR/media-capabilities/" \l "dictdef-mediadecodingconfiguration) *configuration*);

[[NewObject](https://heycam.github.io/webidl/" \l "NewObject)] [Promise](https://heycam.github.io/webidl/#idl-promise)<[MediaCapabilitiesEncodingInfo](https://www.w3.org/TR/media-capabilities/" \l "dictdef-mediacapabilitiesencodinginfo)> *encodingInfo*([MediaEncodingConfiguration](https://www.w3.org/TR/media-capabilities/" \l "dictdef-mediaencodingconfiguration) *configuration*);

};

With the example MediaDecodingType having the following types:

enum *MediaDecodingType* {

["file"](https://www.w3.org/TR/media-capabilities/#dom-mediadecodingtype-file),

["media-source"](https://www.w3.org/TR/media-capabilities/#dom-mediadecodingtype-media-source),

["webrtc"](https://www.w3.org/TR/media-capabilities/#dom-mediadecodingtype-webrtc)

};

And the video configuration currently holding the following properties:

dictionary *VideoConfiguration* {

required [DOMString](https://heycam.github.io/webidl/#idl-DOMString) [contentType](https://www.w3.org/TR/media-capabilities/#dom-videoconfiguration-contenttype);

required [unsigned long](https://heycam.github.io/webidl/#idl-unsigned-long) [width](https://www.w3.org/TR/media-capabilities/#dom-videoconfiguration-width);

required [unsigned long](https://heycam.github.io/webidl/#idl-unsigned-long) [height](https://www.w3.org/TR/media-capabilities/#dom-videoconfiguration-height);

required [unsigned long long](https://heycam.github.io/webidl/#idl-unsigned-long-long) [bitrate](https://www.w3.org/TR/media-capabilities/#dom-videoconfiguration-bitrate);

required [double](https://heycam.github.io/webidl/#idl-double) [framerate](https://www.w3.org/TR/media-capabilities/#dom-videoconfiguration-framerate);

[boolean](https://heycam.github.io/webidl/#idl-boolean) [hasAlphaChannel](https://www.w3.org/TR/media-capabilities/#dom-videoconfiguration-hasalphachannel);

[HdrMetadataType](https://www.w3.org/TR/media-capabilities/#enumdef-hdrmetadatatype) [hdrMetadataType](https://www.w3.org/TR/media-capabilities/#dom-videoconfiguration-hdrmetadatatype);

[ColorGamut](https://www.w3.org/TR/media-capabilities/#enumdef-colorgamut) [colorGamut](https://www.w3.org/TR/media-capabilities/#dom-videoconfiguration-colorgamut);

[TransferFunction](https://www.w3.org/TR/media-capabilities/#enumdef-transferfunction) [transferFunction](https://www.w3.org/TR/media-capabilities/#dom-videoconfiguration-transferfunction);

[DOMString](https://heycam.github.io/webidl/#idl-DOMString) [scalabilityMode](https://www.w3.org/TR/media-capabilities/#dom-videoconfiguration-scalabilitymode);

};

### Conclusion

Media Capabilities in its present form is not useful for VDI, but since this is a new effort, driven by browser vendors (Google, Microsoft), it makes sense to monitor the developments.

## HTMLVideoElement.requestVideoFrameCallback() [4]

<video>.requestVideoFrameCallback() allows web authors to be notified when a frame has been presented for composition.

### Relevance to VDI

This is a new feature which consists of a callback for the video element, that has been decided to be actually implemented (at least for testing) from at least two major browsers (Chrome and Firefox). It fires a callback as soon as a video frame has been decoded and it can be set for single or multiple frames. *It is fired before the requestAnimationFrame callback, which is a window-level callback responsible for refresh the current browser display, as such it makes the decoded frame available prior to being render thus giving time and space for analysis/processing.* Even though in the current version of the spec this is a best-effort feature, thus it might not always be the case. Alongside with the decoded frame (mentioned as “media pixels” in the spec) some metadata are provided as following:

### VideoFrameMetadata

dictionary *VideoFrameMetadata* {

required [DOMHighResTimeStamp](https://www.w3.org/TR/hr-time-2/#dom-domhighrestimestamp) [presentationTime](https://wicg.github.io/video-rvfc/#dom-videoframemetadata-presentationtime);

required [DOMHighResTimeStamp](https://www.w3.org/TR/hr-time-2/#dom-domhighrestimestamp) [expectedDisplayTime](https://wicg.github.io/video-rvfc/#dom-videoframemetadata-expecteddisplaytime);

required [unsigned long](https://heycam.github.io/webidl/#idl-unsigned-long) [width](https://wicg.github.io/video-rvfc/#dom-videoframemetadata-width);

required [unsigned long](https://heycam.github.io/webidl/#idl-unsigned-long) [height](https://wicg.github.io/video-rvfc/#dom-videoframemetadata-height);

required [double](https://heycam.github.io/webidl/#idl-double) [mediaTime](https://wicg.github.io/video-rvfc/#dom-videoframemetadata-mediatime);

required [unsigned long](https://heycam.github.io/webidl/#idl-unsigned-long) [presentedFrames](https://wicg.github.io/video-rvfc/#dom-videoframemetadata-presentedframes);

[double](https://heycam.github.io/webidl/#idl-double) [processingDuration](https://wicg.github.io/video-rvfc/#dom-videoframemetadata-processingduration);

[DOMHighResTimeStamp](https://www.w3.org/TR/hr-time-2/#dom-domhighrestimestamp) [captureTime](https://wicg.github.io/video-rvfc/#dom-videoframemetadata-capturetime);

[DOMHighResTimeStamp](https://www.w3.org/TR/hr-time-2/#dom-domhighrestimestamp) [receiveTime](https://wicg.github.io/video-rvfc/#dom-videoframemetadata-receivetime);

[unsigned long](https://heycam.github.io/webidl/#idl-unsigned-long) [rtpTimestamp](https://wicg.github.io/video-rvfc/#dom-videoframemetadata-rtptimestamp);

};

From the aforementioned metadata it is worth noticing the processingDuration which is defined as following:

*processingDuration*, of type [double](https://heycam.github.io/webidl/#idl-double)

The elapsed duration in seconds from submission of the encoded packet with the same presentation timestamp (PTS) as this frame (e.g. same as the [mediaTime](https://wicg.github.io/video-rvfc/#dom-videoframemetadata-mediatime)) to the decoder until the decoded frame was ready for presentation.

In addition to decoding time, may include processing time. E.g., YUV conversion and/or staging into GPU backed memory.

SHOULD be present. In some cases, user-agents might not be able to surface this information since portions of the media pipeline might be owned by the OS.

### Conclusion

Even if it is just for the processingDuration, this feature can be very useful for adopting VDI in a browser environment since it can give input for reconfiguring the decoders and fine-tuning the decoding process.

## MediaStreamTrack Content Hits [5]

This specification extends [MediaStreamTrack](https://www.w3.org/TR/mediacapture-streams/#dom-mediastreamtrack) to provide a media-content hint attribute. This optional hint permits [MediaStreamTrack](https://www.w3.org/TR/mediacapture-streams/#dom-mediastreamtrack) consumers such as RTCPeerConnection (defined in [[webrtc](https://w3c.github.io/mst-content-hint/" \l "bib-webrtc" \o "WebRTC 1.0: Real-Time Communication Between Browsers)]) or MediaRecorder (defined in [[mediastream-recording](https://w3c.github.io/mst-content-hint/" \l "bib-mediastream-recording" \o "MediaStream Recording)]) to encode or process track media with methods more appropriate to the type of content that is being consumed.

Adding a media-content hint provides a way for a web application to help track consumers make more informed decision of what encoder parameters and processing algorithms to use on the consumed content.

### Relevance to VDI and Conclusion

At its current form this feature is not essential for VDI since it is focusing on the actual content type of the media stream (e.g. for video it can signal “motion”, “detail”, “text”…). We mention it here purely for the sake of completeness because it will probably be adopted at least by Chrome and Firefox.

## WebCodecs [6]

This specification defines interfaces to codecs for encoding and decoding of audio and video.

This specification does not specify or require any particular codec or method of encoding or decoding. The purpose of this specification is to provide JavaScript interfaces to implementations of existing codec technology developed elsewhere. Implementers may support any combination of codecs or none at all.

### Relevance to VDI

This is actually about control of the decoding process (note: in the specification codec is defined as follows: Refers generically to an instance of AudioDecoder, AudioEncoder, VideoDecoder, or VideoEncoder.). As such it mentions that:

### Codec Processing Model

This section is non-normative.

The codec interfaces defined by the specification are designed such that new codec tasks may be scheduled while previous tasks are still pending. For example, web authors may call decode() without waiting for a previous decode() to complete. This is achieved by offloading underlying codec tasks to a separate thread for parallel execution.

This section describes threading behaviors as they are visible from the perspective of web authors. Implementers may choose to use more or less threads as long the externally visible behaviors of blocking and sequencing are maintained as follows.

Regarding the processing model the following are mentioned:

### Control Thread and Codec Thread

All steps in this specification will run on either a [control thread](https://www.w3.org/TR/webcodecs/#control-thread) or a [codec thread](https://www.w3.org/TR/webcodecs/#codec-thread).

The *control thread* is the thread from which authors will construct a [codec](https://www.w3.org/TR/webcodecs/#codec) and invoke its methods. Invoking a codec’s methods will typically result in the creation of [control messages](https://www.w3.org/TR/webcodecs/#control-messages) which are later executed on the [codec thread](https://www.w3.org/TR/webcodecs/#codec-thread). Each [global object](https://html.spec.whatwg.org/multipage/webappapis.html#global-object) has a separate control thread.

The *codec thread* is the thread from which a [codec](https://www.w3.org/TR/webcodecs/#codec) will [dequeue](https://infra.spec.whatwg.org/#queue-dequeue) [control messages](https://www.w3.org/TR/webcodecs/#control-messages) and execute their steps. Each [codec](https://www.w3.org/TR/webcodecs/#codec) instance has a separate codec thread. The lifetime of a codec thread matches that of its associated [codec](https://www.w3.org/TR/webcodecs/#codec) instance.

This will work with interfaces separately for audio and video, with the VideoDecoder Interface (https://www.w3.org/TR/webcodecs/#videodecoder-interface) being obviously the most relevant to this group. The VideoDecoder Interface can be set using VideoDecoderInit (setting output and error handling), parameterized with VideoDecoderConfig and having as input EncodedVideoChuck and as output a VideoFrameOutputCallback. The current configuration parameters are the following:

### VideoDecoderConfig

dictionary *VideoDecoderConfig* {

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

[BufferSource](https://heycam.github.io/webidl/#BufferSource) [description](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-description);

required [unsigned long](https://heycam.github.io/webidl/#idl-unsigned-long) [codedWidth](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-codedwidth);

required [unsigned long](https://heycam.github.io/webidl/#idl-unsigned-long) [codedHeight](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-codedheight);

[unsigned long](https://heycam.github.io/webidl/#idl-unsigned-long) [cropLeft](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-cropleft);

[unsigned long](https://heycam.github.io/webidl/#idl-unsigned-long) [cropTop](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-croptop);

[unsigned long](https://heycam.github.io/webidl/#idl-unsigned-long) [cropWidth](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-cropwidth);

[unsigned long](https://heycam.github.io/webidl/#idl-unsigned-long) [cropHeight](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-cropheight);

[unsigned long](https://heycam.github.io/webidl/#idl-unsigned-long) [displayWidth](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-displaywidth);

[unsigned long](https://heycam.github.io/webidl/#idl-unsigned-long) [displayHeight](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-displayheight);

[HardwareAcceleration](https://www.w3.org/TR/webcodecs/#enumdef-hardwareacceleration) [hardwareAcceleration](https://www.w3.org/TR/webcodecs/#dom-videodecoderconfig-hardwareacceleration) = "allow";

};

## References

1. Media Source Extensions, W3C Draft, <https://www.w3.org/TR/media-source/>
2. Metadata API For Media Resources, W3C Draft, <https://www.w3.org/TR/mediaont-api-1.0>
3. Media Capabilities, W3C Draft, <https://www.w3.org/TR/media-capabilities/>
4. HTMLVideoElement.requestVideoFrameCallback(), W3C Draft, <https://wicg.github.io/video-rvfc/>
5. MediaStreamTrack Content Hits, W3C Draft, <https://w3c.github.io/mst-content-hint/>
6. WebCodecs, W3C Draft, <https://www.w3.org/TR/webcodecs/>

# Khronos Vulkan Video extension (from m57544)

## Proposed design

As seen above, a VK Video Session operates a single decoding session on a single layer. As a result, one can envision two ways of mapping VDI onto Vulkan:

1. Extending a VK Video Session to operate on multiple video decoder instances.
2. Creating a wrapper around multiple VK Video Sessions.

In order to avoid redefining core definitions of the VK Video API, it seems more desirable to opt for option 2 which means mapping a VK Video Session onto a video decoder instance as follows:

Diagram

Description automatically generated

VK Video Session

VK Video Session

Figure - Proposed mapping of a VK Video Session on VDI

Therefore, the VDI concept of grouping video decoding instances will correspond to grouping multiple VK Video Sessions.

## Mapping on Vulkan®

### Overview

Vulkan® Video (VK) is an extension of the Vulkan API which defines standardised functions offered by GPUs. This extension provides interfaces for an application to leverage hardware decoding and encoding capabilities present on GPUs. For more information about the Vulkan API, please refer to [Vulkan-link].

## Mapping of VDI functions

### Overview

The mapping of VDI function on VK is summarised by Table XYZ.

Table XYZ - Summary of VDI function mapping on VK

|  |  |
| --- | --- |
| VDI Functionality | Vulkan mapping |
| **queryCurrent AggregateCapabilities** | New function |
| **getInstance** (grouping) | Extending VkVideoSessionCreateInfoKHR with a group ID attribute. Call of existing vkCreateVideoSessionKHR(). |
| **setConfig (buffer configuration)** | Mapping on existing structures. |
| **getParameter** and **setParameter** | New structure |

### queryCurrentAggregateCapabilities

VkResult vkGetPhysicalDeviceCurrentVideoCapabilitiesMPEG(

VkPhysicalDevice physicalDevice,

**const** VkVideoProfileKHR\* pVideoProfile,

VkCurrentVideoCapabilitiesMPEG\* pCapabilities);

**typedef** **struct** VkCurrentVideoCapabilitiesMPEG {

VkStructureType sType;

**void**\* pNext;

uint32\_t capInstances;

uint64\_t capBufferMemory;

uint32\_t capBitrate;

uint32\_t capMaxSamplesSecond;

VkPerformancePointMPEG capMaxPerformancePoint;

} VkCurrentVideoCapabilitiesMPEG;

**typedef** **struct** VkPerformancePointMPEG {

VkStructureType sType;

**void**\* pNext;

uint32\_t pictureRate;

uint32\_t height;

uint32\_t width;

uint32\_t bitdepth;

} VkPerformancePointMPEG;

### getInstance

VkVideoSessionCreateInfoGroupingMPEG extending the existing VkVideoSessionCreateInfoKHR.

**typedef** **struct** VkVideoSessionCreateInfoGroupingMPEG {

VkStructureType sType;

**const** **void**\* pNext;

uint32\_t groupId;

} VkVideoSessionCreateInfoKHR;

### setConfig

The existing VkVideoSessionCreateInfoKHR and VkVideoPictureResourceKHR map onto setConfig.

**typedef** **struct** VkVideoSessionCreateInfoKHR {

VkStructureType sType;

**const** **void**\* pNext;

uint32\_t queueFamilyIndex;

VkVideoSessionCreateFlagsKHR flags;

**const** VkVideoProfileKHR\* pVideoProfile;

VkFormat pictureFormat;

VkExtent2D maxCodedExtent;

VkFormat referencePicturesFormat;

uint32\_t maxReferencePicturesSlotsCount;

uint32\_t maxReferencePicturesActiveCount;

} VkVideoSessionCreateInfoKHR;

**typedef** **struct** VkVideoProfileKHR {

VkStructureType sType;

**void**\* pNext;

VkVideoCodecOperationFlagBitsKHR videoCodecOperation;

VkVideoChromaSubsamplingFlagsKHR chromaSubsampling;

VkVideoComponentBitDepthFlagsKHR lumaBitDepth;

VkVideoComponentBitDepthFlagsKHR chromaBitDepth;

} VkVideoProfileKHR;

**typedef** **struct** VkVideoPictureResourceKHR {

VkStructureType sType;

**const** **void**\* pNext;

VkOffset2D codedOffset;

VkExtent2D codedExtent;

uint32\_t baseArrayLayer;

VkImageView imageViewBinding;

} VkVideoPictureResourceKHR;

|  |  |
| --- | --- |
| VDI | Vulkan Video |
| sample\_format | pictureFormat |
| sample\_type | pictureFormat |
| sample\_stride | codedExtent |
| line\_stride | codedExtent |
| buffer\_offset | codedOffset |

Note: Mapping to be refined

### getParameter and setParameter

**typedef** **struct** VkVideoSessionOutputParameterMPEG {

VkStructureType sType;

**const** **void**\* pNext;

VkFlag subframeOutput;

void\* metadataCallback;

VkFlag outputCrop;

VkFlag subpictureOutput;

uint32\_t maxOfftimeJitter;

} VkVideoSessionOutputParameterMPEG;

# VDI System Decoder Model (m58863 from MPEG#137)

## Motivations

There have been numerous discussions related to the VDI specification to define clear technical terms specific to the new VDI topics while at the same time fitting well with existing fundamental definition from WG03 Systems such as the term elementary stream.

However, the current status from the draft DIS seems to be halfway in fulfilling the integration with fundamental terms. For instance, the term elementary stream is redefined in VDI in a way that overloads the term and hides the original definition in 14496-1. This definition overloading weakens the integration of the VDI specification into the set of WG03 specification.

Therefore, we suggest that the VDI specification avoids redefining existing terms in order to improve the integration of the VDI specification and future implementation within the set of fundamental WG03 concepts.

## Changes

The changes can be summarized as:

* Removing the definition of elementary stream and access unit and only referring to 14496-1.
* Adding an updated decoder model from 14496-1 to the VDI specification.

Removal of:

3.1

access unit

smallest individually accessible portion of data within an *elementary stream* (3.2) to which unique timing information can be attributed

[SOURCE: ISO/IEC 14496-1]

3.2

elementary stream

consecutive flow of mono-media data consumed by a single decoder

Addition of:

# VDI Systems Decoder Model

## Introduction

The VDI systems decoder model extends on the Systems Decoder model (SDM) defined in ISO/IEC 14496-1. Compared to the SDM, the VDI SDM introduces a new interface in addition to the Elementary Stream Interface called the Media Stream Interface. This interface is the input of the Input Formatter, also called input formatting function, which expects the so-called media streams. The output of the Input Formatter is one or more elementary streams that can be further passed on to the decoders.

These elements are depicted in XYZ.

Shape

Description automatically generated with low confidence

Figure XYZ – VDI Systems Decoder Model

## Concepts of the systems decoder model

This Subclause defines the concepts necessary for the specification of the timing and buffering model. The sequence of definitions corresponds to a walk from the left to the right side of the SDM illustration in Figure 2.

### Media Stream

### Media Stream Interface

The media stream interface is a concept that models the exchange of media stream data and associated between the sync layer and the delivery layer.

### Input formatter

The input formatter takes one or more media streams and generates from them one or more elementary streams. An input formatter may be attached to several decoding buffers when it produces individual elementary streams or multi-layer elementary streams.

### Access Units (AU)

*(Copy of 14496-1, will be replaced by see 14496-1)*

Elementary stream data is partitioned into access units. The delineation of an access unit is completely determined by the entity that generates the elementary stream (e.g., the compression layer). An access unit is the smallest data entity to which timing information can be attributed. Two access units from the same elementary stream shall never refer to the same decoding or composition time. Any further partitioning of the data in an elementary stream is not visible for the purposes of the systems decoder model. Access units are conveyed by SL-packetized streams and are received by the decoding buffers. The decoders consume access units with the necessary side information (e.g., time stamps) from the decoding buffers.

NOTE — An ISO/IEC 14496-1 compliant terminal implementation is not required to process each incoming access unit as a whole. It is furthermore possible to split an access unit into several fragments for transmission as specified in ‎7.3. This allows the sending terminal to dispatch partial AUs immediately as they are generated during the encoding process. Such partial AUs may have significance for improved error resilience.

### Decoding Buffer (DB)

*(Copy of 14496-1, will be replaced by see 14496-1)*

The decoding buffer is a buffer at the input of an elementary stream decoder in the receiving terminal that receives and stores access units. The systems buffer model enables the sending terminal to monitor the decoding buffer resources that are used during a presentation.

### Elementary Streams (ES)

*(Copy of 14496-1, will be replaced by see 14496-1)*

Streaming data received at the output of a decoding buffer, independent of its content, is considered as an elementary stream for the purpose of ISO/IEC 14496. The elementary streams are produced and consumed by the compression layer entities (encoders and decoders, respectively). ISO/IEC 14496 assumes that the integrity of an elementary stream is preserved from end to end.

### Elementary Stream Interface (ESI)

*(Copy of 14496-1, will be replaced by see 14496-1)*

The elementary stream interface is a concept that models the exchange of elementary stream data and associated control information between the compression layer and the sync layer. It is explained further in ‎7.3.

### Decoder

*(Copy of 14496-1, will be replaced by see 14496-1)*

For the purposes of this model, the decoder extracts access units from the decoding buffer at precisely defined points in time and places composition units, the results of the decoding processes, in the composition memory. A decoder may be attached to several decoding buffers.

### Composition Units (CU)

*(Copy of 14496-1, will be replaced by see 14496-1)*

Decoders consume access units and produce composition units. An access unit corresponds to an integer number of composition units. In case of multiple elementary streams attached to a single decoder (scalable coding), each composition unit is derived from access units from one or more of these streams. Composition units reside in composition memory.

### Composition Memory (CM)

*(Copy of 14496-1, will be replaced by see 14496-1)*

The composition memory is a random access memory that contains composition units. The size of this memory is not normatively specified.

### Compositor

*(Copy of 14496-1, will be replaced by see 14496-1)*

The compositor takes composition units out of the composition memory and either consumes them (e.g., composes and presents them, in the case of audio-visual data) or skips them. The compositor is not specified in ISO/IEC 14496-1, as the details of this operation are not relevant within the context of the systems decoder model. ‎7.1.3.5 defines which composition units are available to the compositor at any instant of time.

*End of proposed new text.*

# Layer composite position info SEI message (from JVET-S0107)

## Layer composite position info SEI message

|  |  |
| --- | --- |
| layer\_composite\_position\_info( ) { | Descriptor |
| **lcpi\_param\_num\_bits\_minus1** | u(12) |
| **lcpi\_top\_left\_ pos\_in\_units\_ver**[ nuh\_layer\_id] | u(v) |
| **lcpi\_top\_left\_pos\_in\_units\_hor**[ nuh\_layer\_id] | u(v) |
| **lcpi\_height\_in\_units**[ nuh\_layer\_id] | u(v) |
| **lcpi\_width\_in\_units**[ nuh\_layer\_id] | u(v) |
| **}** |  |

The layer composite position info SEI message describes the recommended position and size of the decoded picture of the current layer within a recommended composite picture comprised of decoded pictures from multiple layers.

**lcpi\_param\_num\_bits\_minus1** + 1 specifies the number of bits used to represent the lcpi\_top\_left\_pos\_ver[ nuh\_layer\_id ], cpi\_top\_left\_pos\_ver[ nuh\_layer\_id ], lcpi\_width[ nuh\_layer\_id ], and lcpi\_height[ nuh\_layer\_id ] syntax elements.

**lcpi\_top\_left\_pos\_ver**[ nuh\_layer\_id]and **lcpi\_top\_left\_pos\_ver**[ nuh\_layer\_id]indicate the recommended composite display vertical and horizontal positions, respectively, for the decoded picture of the current layer. The number of bits to represent the syntax elements is lcpi\_param\_num\_bits\_minus1 + 1.

**lcpi\_width**[ nuh\_layer\_id]and **lcpi\_height**[ nuh\_layer\_id] indicate the recommended composite display width and height, respectively, for the decoded picture of the current layer. The number of bits to represent the syntax elements is lcpi\_param\_num\_bits\_minus1 + 1.

## Recommended composite layers info SEI message

|  |  |
| --- | --- |
| recommended\_composite\_layers\_info( ) { | Descriptor |
| **rcli\_cancel\_flag** | u(1) |
| if (!rcli\_cancel\_flag) { |  |
| **rcli\_persistence\_flag** | u(1) |
| **rcli\_layer\_scaling\_enabled\_flag** | u(1) |
| **rcli\_layer\_overlap\_enabled\_flag** | u(1) |
| **rcli\_unit\_size\_present\_flag** | u(1) |
| **rcli\_composite\_size\_present\_flag** | u(1) |
| **rcli\_offset\_present\_flag** | u(1) |
| **num\_olss\_minus1** | u(8) |
| for ( i = 0; i <= num\_olss\_minus1; i++ ) { |  |
| if (rcli\_unit\_size\_present\_flag ) { |  |
| **rcli\_unit\_size\_ver**[ i ] | u(16) |
| **rcli\_unit\_size\_hor**[ i ] | u(16) |
| } |  |
| if (rcli\_composite\_size\_present\_flag ) { |  |
| **rcli\_composite\_size\_ver**[ i ] | u(16) |
| **rcli\_composite\_size\_hor**[ i ] | u(16) |
| } |  |
| if ( rcli\_offset\_present\_flag ) { |  |
| **rcli\_offset\_ver**[ i ] | s(16) |
| **rcli\_offset\_hor**[ i ] | s(16) |
| } |  |
| } |  |
| } |  |
| } |  |

The recommended composite layers info SEI message, with the layer composite position information SEI message, describes a layout of decoded pictures from the layers of an OLS within a recommended composite picture, according to the composition process of sub-clause X.

**rcli\_cancel\_flag** equal to 1 indicates that the SEI message cancels the persistence of any previous recommended composite layers info SEI message in output order.

**rcli\_persistence\_flag** specifies the persistence of recommended composite layers info SEI message.

rcli\_persistence\_flag equal to 0 specifies that the recommended composite layers info SEI message applies to the current AU only.

rcli\_persistence\_flag equal to 1 specifies that the recommended composite layers info SEI message applies to the current access unit and persists all subsequent access units in output order until one or more of the following conditions are true:

* A new CVS begins.
* The bitstream ends.
* A picture in an AU associated with a recommended composite layers info SEI message is output that follows the current picture in output order

**rcli\_layer\_scaling\_enabled\_flag** equal to 0 indicates that layerPicSizeInCompositeHeight[ i ] and layerPicSizeInCompositeWidth[ i ] derived in subclause X are equal to the width and height, respectively, of the coded picture with nuh\_layer\_id equal to i. rcli\_layer\_scaling\_enabled\_flag equal to 1 indicates that layerPicSizeInCompositeHeight[ i ] and layerPicSizeInCompositeWidth[ i ] may differ from the width and height, respectively, of the coded picture with nuh\_layer\_id equal to i.

**rcli\_layer\_overlap\_enabled\_flag** equal to 0 indicates that all values of Count[ y][ x ] shall be les than or equal to 1, as derived in subclause X . rcli\_layer\_overlap\_enabled\_flag equal to 1 does not impose a restriction.

**rcli\_unit\_size\_present\_flag** equal to 1 specifies that the rcli\_unit\_size\_ver[ i ] and rcli\_ unit\_size\_hor[ i ] syntax elements are present. rcli\_unit\_size\_present\_flag equal to 0 specifies that the rcli\_ unit\_size\_ver[ i ] and rcli\_ unit\_size\_hor[ i ] syntax elements are not present.

**rcli\_composite\_size\_present\_flag** equal to 1 specifies that the rcli\_composite\_size\_ver[ i ] and rcli\_composite\_size\_hor[ i ] syntax elements are present. rcli\_composite\_size\_present\_flag equal to 0 specifies that the rcli\_composite\_size\_ver[ i ] and rcli\_composite\_size\_hor[ i ] syntax elements are not present.

**rcli\_offset\_present\_flag** equal to 1 specifies that the rcli\_offset\_ver[ i ] and rcli\_offset\_hor[ i ] syntax elements are present. rcli\_offset\_present\_flag equal to 0 specifies that the rcli\_offset\_ver[ i ] and rcli\_offset\_hor[ i ] syntax elements are not present.

**num\_olss\_minus1** indicates the number of OLSs for which syntax elements are present in the SEI message.

**rcli\_unit\_size\_ver**[ i ] and **rcli\_unit\_size\_hor**[ i ] indicate vertical and horizontal unit size parameters respectively, used in the composition process in subclause X for the i-th OLS. When not present, the values of rcli\_offest\_ver[ i ] and rcli\_offset\_hor[ i ] may be determined by external means.

**rcli\_composite\_size\_ver**[ i ] and **rcli\_composite\_size\_hor**[ i ] indicate the vertical and horizontal size, respectively, of the recommended composite picture in luma samples used in the composition process in subclause X for the i-th OLS. When not present, the values of rcli\_offest\_ver[ i ] and rcli\_offset\_hor[ i ] may be determined by external means.

**rcli\_offest\_ver**[ i ]and **rcli\_offset\_hor**[ i ]indicate vertical and horizontal offsets, respectively, of the positions of the decoded layer pictures used in the composition process in subclause X for the i-th OLS. When not present, the values of rcli\_offest\_ver[ i ] and rcli\_offset\_hor[ i ] are inferred to be equal to 0.

## Recommended composition process

This subclause describes a composition process to derive sample values for a recommended composite picture, CompositePicture, for the i-th OLS.

CompositePicture[ 0]  is a 2-D sample array of size rcli\_composite\_size\_hor x rcli\_composite\_size\_ver, of the luma samples of CompositePicture. CompositePicture[ cIdx ]  for cIdx in 1 .. 2 are 2-D sample arrays of size rcli\_composite\_size\_hor/ SubWidthC x rcli\_composite\_size\_ver/ SubHeightC.

For each layer j included in the i-th OLS, if a picture is present in the AU with nuh\_layer\_id equal to j, the recommended size of the representation of the decoded picture in the composite picture, scaledLayerPic[ j ], is layerPicSizeInCompositeWidth[j ] x layerPicSizeInCompositeHeight[ j ], in luma samples, as derived below.

layerPicSizeInCompositeHeight[ j ] = lcpi\_height\_in\_units[j ] \* rcli\_unit\_size\_ver[ i ]

layerPicSizeInCompositeWidth[ j ] = lcpi\_width\_in\_units[ j ] \* rcli\_unit\_size\_hor[ i ]

When rcli\_layer\_scaling\_enabled\_flag equal to 0, scaledLayerPic[ j ] is set to the decoded picture.

Otherwise, scaledLayerPic[ j ] is derived by scaling the decoded picture.

scaledLayerPic[ j ] is a picture of size layerPicSizeInCompositeWidth[ j ] x layerPicSizeInCompositeHeight[ j ], in the luma samples .

The sample values of CompositePicture are derived as follows:

for (y = 0; y < rcli\_composite\_size\_ver[ i]; y++)

for (x = 0; x < rcli\_composite\_size\_hor[ i ]; x++)

Count[ y ][ x ] = 0

for (j = 0 ; j< 64; j++)

if (j is in the i-th OLS && a picture is present in the AU with nuh\_layer\_id = j)

comp\_y = rcli\_offset\_ver[ i ] + lcpi\_top\_left\_pos\_in\_units\_ver[ j ] \* rcli\_unit\_size\_ver

comp\_x = rcli\_offset\_hor[ i ] + lcpi\_top\_left\_pos\_in\_units\_hor[ j ] \* rcli\_unit\_size\_hor

for (y = 0; y < layerPicSizeInCompositeHeight[ j ] \* ; y++)

for (x = 0; x < layerPicSizeInCompositeWidth[ j ]; x++)

CompositePicture[ 0 ] [ comp\_y + y ][ comp\_x + x ] = scaledLayerPic[ 0 ][ j ][ y ][ x ]

Count[[ comp\_y + y ][ comp\_x + x ]]++

for (y = 0; y < layerPicSizeInCompositeHeight[ j ]/SubWidth \* ; y++)

for (x = 0; x < layerPicSizeInCompositeWidth[ j ]/SubHeight; x++)

CompositePicture[ 1 ][ comp\_y/SubHeight + y ][ comp\_x/SubWidth + x ] =

scaledLayerPic[ 1 ][ j ][ y ][ x ]

CompositePicture[ 2 ][ comp\_y/SubHeight + y ][ comp\_x/SubWidth + x ] =

scaledLayerPic[ 2 ][ j ][ y ][ x ]

The value of samples of CompositePicture[cIdx] not assigned above are undefined.

# Composite picture information SEI message (from JVET-T0049)

## Syntax of CPI SEI message

|  |  |
| --- | --- |
| composite\_picture\_info( ) { | Descriptor |
| **cpi\_position\_and\_size\_bit\_len** | ue(v) |
| **cpi\_num\_patches\_minus1** | ue(v) |
| **cpi\_cropped\_source\_area\_allowed\_flag** | u(1) |
| **cpi\_scaling\_allowed\_flag** | u(1) |
| **cpi\_gap\_allowed\_flag** | u(1) |
| for ( i = 0; i <= cpi\_num\_patches\_minus1; i++ ) { |  |
| **cpi\_patch\_layer\_id**[ i ] | u(6) |
| if( cpi\_cropped\_source\_area\_allowed\_flag ) { |  |
| **cpi\_patch\_source\_top\_left\_x**[ i ] | u(v) |
| **cpi\_patch\_source\_top\_left\_y**[ i ] | u(v) |
| } |  |
| **cpi\_patch\_source\_luma\_width\_minus1**[ i ] | u(v) |
| **cpi\_patch\_source\_luma\_height\_minus1**[ i ] | u(v) |
| **cpi\_patch\_dest\_top\_left\_x**[ i ] | u(v) |
| **cpi\_patch\_dest\_top\_left\_y**[ i ] | u(v) |
| if( cpi\_scaling\_flag) { |  |
| **cpi\_patch\_dest\_luma\_width\_minus1**[ i ] | u(v) |
| **cpi\_patch\_dest\_luma\_height\_minus1**[ i ] | u(v) |
| } |  |
| } |  |
| } |  |

## Semantics of CPI SEI message

A composite picture info (CPI) SEI message provides information for constructing composite picture from one or more patches originated from pictures of an access unit.

When one or more CPI SEI message is present in a bitstream, the first CPI SEI message shall be present in a CVSS AU. When two or more CPI SEI message are present in an AU, the content of the CPI SEI messages shall be the same.

When a scalable nesting SEI message contains a CPI SEI message, the value of sn\_ols\_flag shall be equal to 1.

A CPI SEI message applies to the AU A that contains the SEI message and all AUs that follow the AU A in output order until one of the following applies:

* The end of the bitstream.
* The next AU contains a new CPI SEI message that applies to the target OLS.

When present, CPI SEI message or scalable nesting SEI message that contains CPI SEI message, shall be contained in an access unit with temporal Id equal to 0. The temporal Id of the CPI SEI message or the scalable nesting SEI message that contains the CPI SEI message shall be equal to 0.

**cpi\_position\_and\_size\_bit\_len\_minus1** plus 1 specifies the number of bits for signalling of syntax elements cpi\_width\_in\_luma\_samples, cpi\_height\_in\_luma\_samples, cpi\_patch\_source\_top\_left\_x[ i ], cpi\_patch\_source\_top\_left\_y[ i ], cpi\_patch\_source\_width\_minus1[ i ] , cpi\_patch\_source\_height\_minus1[ i ], cpi\_patch\_dest\_top\_left\_x[ i ], cpi\_patch\_dest\_top\_left\_y[ i ], cpi\_patch\_source\_dest\_minus1[ i ], and cpi\_patch\_source\_dest\_minus1[ i ].

**cpi\_num\_patches\_minus1** specifies the number of patches of each composite picture created from AUs in which the SEI applies.

**cpi\_cropped\_source\_area\_allowed\_flag** equal to 1 specifies that cpi\_patch\_source\_top\_left\_x[ i ] and cpi\_patch\_source\_top\_left\_y[ i ] are present. cpi\_cropped\_source\_area\_allowed\_flag equal to 0 specifies that cpi\_patch\_source\_top\_left\_x[ i ] and cpi\_patch\_source\_top\_left\_y[ i ] are present are not present.

**cpi\_scaling\_allowed\_flag** equal to 1 specifies that cpi\_patch\_dest\_luma\_width[ i ] and cpi\_patch\_dest\_luma\_width[ i ] are present. cpi\_scaling\_allowed\_flag equal to 0 specifies that cpi\_patch\_dest\_luma\_width[ i ] and cpi\_patch\_dest\_luma\_width[ i ] are not present.

**cpi\_gap\_allowed\_flag** equal to 1 specifies that the union of all patches does not cover all area in the composite picture. cpi\_gap\_allowed\_flag equal to 0 specifies that the union of all patches covers all area in the composite picture.

NOTE: When cpi\_gap\_allowed\_flag is equal to 1, pels in the area that is not covered by any patch need to be initialized with a valid pel value.

**cpi\_patch\_layer\_id**[ i ] specifies the layer Id of the source picture for the i-th patch.

It is constrained that when CPI SEI message is contained in a scalable nesting SEI message, the value of cpi\_patch\_layer\_id[ i ] for i in the ranges from 0 to cpi\_num\_patches\_minus1, inclusive, shall be equal to one of layer included in the OLS associated with the SEI message.

It is constrained that when CPI SEI message is not contained in a scalable nesting SEI message, the value of cpi\_patch\_layer\_id[ i ] for i in the ranges from 0 to cpi\_num\_patches\_minus1, inclusive, shall be equal to one of layer that is present in the bitstream.

It is constrained that the layer with layer Id equal to cpi\_patch\_layer\_id[ i ] for i in the ranges from 0 to cpi\_num\_patches\_minus1, inclusive, shall be an output layer in the output layer set associated with the SEI message.

Picture with layer Id equal to cpi\_patch\_layer\_id[ i ] for i in the ranges from 0 to cpi\_num\_patches\_minus1, inclusive, may not be present in the AU in which the SEI applies. When The AU has no picture in cpi\_patch\_layer\_id[ i ], application that generate the composite picture based on the SEI may assign any valid pel value for the i-th patch in the composite picture. Furthermore, in such situation, application may further assign the same pel values for the entire area covered the i-th patch.

**cpi\_patch\_source\_top\_left\_x**[ i ] specifies horizontal position of top left pel in the source picture for the i-th patch. The value of cpi\_patch\_source\_top\_left\_x[ i ] shall be less than the width of the source picture minus 1 (or alternatively, minus 2). The length of the cpi\_patch\_source\_top\_left\_x[ i ] syntax element is cpi\_position\_and\_size\_bit\_len\_minus1 + 1 bits. When not present, the value of cpi\_patch\_source\_top\_left\_x[ i ] is inferred to be equal to 0.

**cpi\_patch\_source\_top\_left\_y**[ i ] specifies vertical position of top left pel in the source picture for the i-th patch. The value of cpi\_patch\_source\_top\_left\_y[ i ] shall be less than the height of the source picture minus 1 (or alternatively, minus 2). The length of the cpi\_patch\_source\_top\_left\_y[ i ] syntax element is cpi\_position\_and\_size\_bit\_len\_minus1 + 1 bits. When not present, the value of cpi\_patch\_source\_top\_left\_y[ i ] is inferred to be equal to 0.

**cpi\_patch\_source\_width\_minus1**[ i ] plus 1 specifies width of the area in the source picture for the i-th patch. The sum of cpi\_patch\_source\_top\_left\_x[ i ] and cpi\_patch\_source\_width\_minus1[ i ] shall be less than the width, in the unit of luma sample, of the source picture. The length of the cpi\_patch\_source\_width\_minus1[ i ] syntax element is cpi\_position\_and\_size\_bit\_len\_minus1 + 1 bits.

When the value of cpi\_cropped\_source\_area\_allowed\_flag equal to 0, it is constrained that the value of cpi\_patch\_source\_width\_minus1[ i ] shall be equal to the pps\_pic\_width\_in\_luma\_samples − 1 of the associated picture.

**cpi\_patch\_source\_height\_minus1**[ i ] plus 1 specifies height of the area in the source picture for the i-th patch. The sum of cpi\_patch\_source\_top\_left\_y[ i ] and cpi\_patch\_source\_height\_minus1[ i ] shall be less than the height, in the unit of luma sample, of the source picture. The length of the cpi\_patch\_source\_height\_minus1[ i ] syntax element is cpi\_position\_and\_size\_bit\_len\_minus1 + 1 bits.

When the value of cpi\_cropped\_source\_area\_allowed\_flag equal to 0, it is constrained that the value of cpi\_patch\_source\_height\_minus1[ i ] shall be equal to the pps\_pic\_height\_in\_luma\_samples − 1 of the associated picture.

The value of cpi\_patch\_source\_top\_left\_x[ i ] + cpi\_patch\_source\_width\_minus1[ i ] shall be less than the picture width in luma samples minus 1 of any picture in the layer with layer id equal to cpi\_patch\_layer\_id[ i ] which the SEI message applies.

The value of cpi\_patch\_source\_top\_left\_y[ i ] + cpi\_patch\_source\_height\_minus1[ i ] shall be less than the picture height in luma samples minus 1 of any picture in the layer with layer id equal to cpi\_patch\_layer\_id[ i ] which the SEI message applies.

**cpi\_patch\_dest\_top\_left\_x**[ i ] specifies horizontal position of top left pel in the composite picture for the i-th patch. The value of cpi\_patch\_dest\_top\_left\_x[ i ] shall be less than the value of cpi\_width\_in\_luma\_samples – 1 (or alternatively, minus 2). The length of the cpi\_patch\_dest\_top\_left\_x[ i ] syntax element is cpi\_position\_and\_size\_bit\_len\_minus1 + 1 bits.

**cpi\_patch\_dest\_top\_left\_y**[ i ] specifies vertical position of top left pel in the composite picture for the i-th patch. The value of cpi\_patch\_dest\_top\_left\_y[ i ] shall be less than the value of cpi\_height\_in\_luma\_samples – 1 (or alternatively, minus 2). The length of the cpi\_patch\_dest\_top\_left\_y[ i ] syntax element is cpi\_position\_and\_size\_bit\_len\_minus1 + 1 bits.

**cpi\_patch\_dest\_width\_minus1**[ i ] plus 1 specifies width of the area in the composite picture for the i-th patch. The sum of cpi\_patch\_dest\_top\_left\_x[ i ] and cpi\_patch\_dest\_width\_minus1[ i ] shall be less than the value of cpi\_width\_in\_luma\_samples – 1. When not present, the value of cpi\_patch\_dest\_width\_minus1[ i ] is inferred to be equal to cpi\_source\_width\_minus1[ i ]. The length of the cpi\_patch\_dest\_width\_minus1[ i ] syntax element is cpi\_position\_and\_size\_bit\_len\_minus1 + 1 bits.

**cpi\_patch\_dest\_height\_minus1**[ i ] plus 1 specifies height of the area in the composite picture for the i-th patch. The sum of cpi\_patch\_dest\_top\_left\_y[ i ] and cpi\_patch\_dest\_height\_minus1[ i ] shall be less than the value of cpi\_height\_in\_luma\_samples – 1. When not present, the value of cpi\_patch\_dest\_height\_minus1[ i ] is inferred to be equal to cpi\_source\_height\_minus1[ i ]. The length of the cpi\_patch\_dest\_height\_minus1[ i ] syntax element is cpi\_position\_and\_size\_bit\_len\_minus1 + 1 bits.

The variables CompositePictureWidth and CompositePictureHeight, specifying the width and height, respectively, of the composite picture are derived as follows:

CompositePictureWidth = cpi\_patch\_dest\_top\_left\_x[ 0 ] + cpi\_patch\_dest\_width\_minus1[ 0 ] + 1  
CompositePictureHeight = cpi\_patch\_dest\_top\_left\_y[ 0 ] + cpi\_patch\_dest\_height\_minus1[ 0 ] + 1  
for( i = 0; i <= cpi\_num\_patches\_minus1; i++ ) {  
 if( (cpi\_patch\_dest\_top\_left\_x[ i ] + cpi\_patch\_dest\_width\_minus1[ i ] + 1 ) > CompositePictureWidth )  
 CompositePictureWidth = cpi\_patch\_dest\_top\_left\_x[ i ] + cpi\_patch\_dest\_width\_minus1[ i ] + 1  
 if( (cpi\_patch\_dest\_top\_left\_y[ i ] + cpi\_patch\_dest\_height\_minus1[ i ] + 1 ) > CompositePictureHeight )  
 CompositePictureHeight = cpi\_patch\_dest\_top\_left\_y[ i ] + cpi\_patch\_dest\_height\_minus1[ i ] + 1  
}

For area that is covered by two or more patches, the value of each pel in such area shall be from the patch with the highest i-th index among those patches.

It is constrained that there shall be no two patches in the composite picture, i-th patch and j-th patch, such that the values of cpi\_patch\_dest\_top\_left\_x[ i ], cpi\_patch\_dest\_top\_left\_y[ i ], cpi\_patch\_dest\_width\_minus1[ i ], and cpi\_patch\_dest\_height\_minus1[ i ] are equal to cpi\_patch\_dest\_top\_left\_x[ j ], cpi\_patch\_dest\_top\_left\_y[ j ], cpi\_patch\_dest\_width\_minus1[ j ], and cpi\_patch\_dest\_height\_minus1[ j ], respectively.

It is constrained that when two patches in the composite picture, i-th patch and the j-th patch, have the values of cpi\_patch\_dest\_top\_left\_x[ i ] and cpi\_patch\_dest\_top\_left\_y[ i ] equal to cpi\_patch\_dest\_top\_left\_x[ j ] and cpi\_patch\_dest\_top\_left\_y[ j ], respectively, and the values of cpi\_patch\_dest\_width[ i ] and cpi\_patch\_dest\_height[ i ] are greater than cpi\_patch\_dest\_width[ j ] and cpi\_patch\_dest\_height[ j ], respectively, then j shall be greater than i.