**ISO/IEC 23090-13:222x(E)**

ISO TC 1 SC 29/WG 03

Secretariat: XXXX

**Information technology — Coded representation of immersive media — Part 13: Video decoding interface for immersive media**

FDIS stage

**Warning for WDs and CDs**

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

© ISO 2020

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO’s member body in the country of the requester.

ISO copyright office

CP 401 • Ch. de Blandonnet 8

CH-1214 Vernier, Geneva

Phone: +41 22 749 01 11

Fax: +41 22 749 09 47

Email: copyright@iso.org

Website: www.iso.org

Published in Switzerland

Contents

[Foreword v](#_Toc120279128)

[Introduction vi](#_Toc120279129)

[1 Scope 1](#_Toc120279130)

[2 Normative references 1](#_Toc120279132)

[3 Terms and definitions 1](#_Toc120279134)

[4 Abbreviations 2](#_Toc120279135)

[5 Video decoding engine 2](#_Toc120279140)

[5.1 General 2](#_Toc120279141)

[5.2 Input video decoding interface 4](#_Toc120279142)

[5.3 Output video decoding interface 4](#_Toc120279143)

[5.4 Control interface to the video decoding interface 5](#_Toc120279144)

[5.4.1 Functions 5](#_Toc120279145)

[5.5 Examples of video decoding engine instantiations 10](#_Toc120279146)

[5.5.1 Mapping on OpenMAX™ Integration Layer (OpenMAX IL) 10](#_Toc120279147)

[5.5.2 Mapping on Vulkan® Video 10](#_Toc120279148)

[5.5.3 Informative mapping 13](#_Toc120279149)

[6 VDI systems decoder model 13](#_Toc120279150)

[6.1 Introduction 13](#_Toc120279151)

[6.2 Concepts of the VDI systems decoder model 14](#_Toc120279152)

[6.2.1 General 14](#_Toc120279153)

[6.2.2 Media stream 14](#_Toc120279154)

[6.2.3 Media stream interface 14](#_Toc120279155)

[6.2.4 Input formatter 14](#_Toc120279156)

[6.2.5 Access Units (AU) 14](#_Toc120279157)

[6.2.6 Decoding Buffer (DB) 14](#_Toc120279158)

[6.2.7 Elementary Streams (ES) 15](#_Toc120279159)

[6.2.8 Elementary Stream Interface (ESI) 15](#_Toc120279160)

[6.2.9 Decoder 15](#_Toc120279161)

[6.2.10 Composition Units (CU) 15](#_Toc120279162)

[6.2.11 Composition Memory (CM) 15](#_Toc120279163)

[6.2.12 Compositor 15](#_Toc120279164)

[7 Video decoder interface 15](#_Toc120279165)

[7.1 General 15](#_Toc120279166)

[7.2 Operations on input media streams 15](#_Toc120279167)

[7.2.1 General 15](#_Toc120279168)

[7.2.2 Concepts 16](#_Toc120279173)

[7.2.3 Filtering by video object identifier 16](#_Toc120279174)

[7.2.4 Inserting video objects 17](#_Toc120279175)

[7.2.5 Appending two video objects 18](#_Toc120279176)

[7.2.6 Stacking two video objects 19](#_Toc120279177)

[7.3 Slice-based instantiation for ISO/IEC 23008-2 High efficiency video coding (HEVC) 20](#_Toc120279178)

[7.3.1 General 20](#_Toc120279179)

[7.3.2 Media and elementary stream constraints 20](#_Toc120279180)

[7.4 Layer-based instantiation for ISO/IEC 23090-3 Versatile video coding (VVC) 21](#_Toc120279183)

[7.4.1 General 21](#_Toc120279184)

[7.4.2 Media and elementary stream constraints 21](#_Toc120279185)

[7.5 Slice-based instantiation for ISO/IEC 23094-1 Essential video coding (EVC) 23](#_Toc120279186)

[7.5.1 General 23](#_Toc120279187)

[7.5.2 Media and elementary streams constraints 24](#_Toc120279188)

[Annex A (normative) Control interface IDL definition 26](#_Toc120279189)

[A.1 General 26](#_Toc120279190)

[Annex B (informative) OpenMAX IL VDI extension header 27](#_Toc120279191)

[B.1 General 27](#_Toc120279192)

[Annex C (normative) Supplemental enhancement information (SEI) syntax and semantics 28](#_Toc120279193)

[C.1 VDI SEI envelope 28](#_Toc120279194)

[C.2 Independent layer info SEI message 28](#_Toc120279195)

[C.3 Examples of video object positioning (informative) 31](#_Toc120279196)

[C.3.1 Appending 31](#_Toc120279197)

[C.3.2 Appending and stacking 32](#_Toc120279198)

[Annex D (informative) Example implementations of input formatting operations 35](#_Toc120279199)

[D.1 General 35](#_Toc120279200)

[D.2 Creating a 2-by-2 video mosaic via application control 35](#_Toc120279201)

[D.3 Creating a 2-by-2 video mosaic via SEI control 37](#_Toc120279202)

[Annex E (informative) Brief description of OpenMAX IL functions 39](#_Toc120279203)

[E.1 Decoder Engine Control Interface 39](#_Toc120279204)

[E.1.1 OMX\_Init() and OMX\_Deinit() 39](#_Toc120279205)

[E.1.2 OMX\_GetHandle() and OMX\_FreeHandle() 39](#_Toc120279206)

[E.1.3 OMX\_SetupTunnel() and OMX\_TeardownTunnel() 39](#_Toc120279207)

[E.2 Decoder Instance Interface 39](#_Toc120279208)

[E.2.1 Methods 39](#_Toc120279209)

[E.2.2 Media Input and Output Interface 40](#_Toc120279210)

[E.2.3 Format of the OpenMAX IL buffer header 41](#_Toc120279211)

[E.2.4 Buffer flags defined in OpenMAX IL 41](#_Toc120279212)

[E.2.5 Input/Output from/into GPU 41](#_Toc120279226)

[Annex F (informative) Mapping on Media Source Extensions (MSE) 42](#_Toc120279227)

[F.1 Overview 42](#_Toc120279228)

[F.2 Mapping of VDI functions 43](#_Toc120279229)

[Bibliography 44](#_Toc120279230)

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see [www.iso.org/directives](https://www.iso.org/directives-and-policies.html)).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see [www.iso.org/patents](https://www.iso.org/iso-standards-and-patents.html)).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see [www.iso.org/iso/foreword.html](https://www.iso.org/foreword-supplementary-information.html).

This document was prepared by Technical Committee ISO/IEC/JTC 1 *Information technology*, Subcommittee SC 29, Coding of audio, picture, multimedia and hypermedia information.

This is the first edition published under ISO/IEC 23090-13:222x.

A list of all parts in the ISO 23090 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at [www.iso.org/members.html](https://www.iso.org/members.html).

Introduction

The interfaces and operations specified in this document come as extensions of existing video decoding engine specifications exposing hardware video decoding capabilities.

Information technology — Coded representation of immersive media — Part 13: Video decoding interface for immersive media

# Scope

This document specifies the interfaces of a video decoding engine as well as the operations related to elementary streams and metadata that can be performed by this video decoding engine . To support those operations, this document also specifies SEI messages when necessary for certain video codecs.

# Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 23008-2, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 2: High efficiency video coding*

ISO/IEC 23090-3, *Information technology — Coded representation of immersive media — Part 3: Versatile video coding*

ISO/IEC 23094-1, *Information technology — General video coding — Part 1: Essential video coding*

# Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1

media stream

part of an *elementary stream* (3.2) or one or more aggregated *elementary streams* (3.2)

NOTE 1 to entry: Every elementary stream is a media stream, but the inverse is not true.

NOTE 2 to entry: A media stream may contain metadata such as non-VCL NAL units.

3.2

subframe

independently decodable unit smaller than a frame to which post-decoding processing by the decoder, if any, has been applied

3.3

video object

independently decodable substream of a video *elementary stream* (3.2)

3.4

video object identifier

an integer identifying a *video object* (3.4)

# Abbreviations

API application programming interface

ES elementary stream

I video object identifier

IVDI input video decoding interface

NAL network abstraction layer

OLS Output Layer Set

OVDI output video decoding interface

PPS picture parameter set

SEI supplemental enhancement information

SPS sequence parameter set

VCL video coding layer

VDE video decoding engine

MDS media stream

# Video decoding engine

## General

The Video Decoding Engine (VDE) enables the decoding, the synchronization and the formatting of associated media streams into elementary streams. The media streams are fed through the Input Video Decoding Interface (IVDI) of the VDE and provided to the subsequent elements of the rendering pipeline via the Output Video Decoding Interface (OVDI) in their decoded form. Between the input and the output, the VDE may extract and/ore merge independently decodable regions from a set of input media streams via the input formatting function to generate a set of elementary streams that are fed into the video decoder instances, which run inside the engine. The VDE can execute a merging operation or an extraction operation on the input media streams such that the number of running video decoder instances is different from the number of input media streams that are required by the application. For example, a VDE might not be capable of decoding a single 4K input media stream with one decoder instance, but it might be able to decode some of the independently decodable regions, at a lower resolution, present in that input media stream. In this case, the VDE should verify the availability of sufficient resources to run those video decoder instances in parallel.

Figure 1 represents the architecture for the VDE and the associated IVDI and OVDI interfaces.



**Key**

|  |  |  |  |
| --- | --- | --- | --- |
| MDS | media stream |  |  |
| ES | elementary stream |  |  |
| MTS | metadata stream |  |  |
| DS | decoded sequence |  |  |
| m | number of input metadata streams |  |  |
| n | number of media streams |  |  |
| j | number of video decoder instances |  |  |
| p | number of output metadata streams |  |  |
| q | number of decoded sequences |

Figure 1 — Video decoding engine and interfaces

NOTE Multiple elementary streams that are output of the input formatting function can be fed to a single video decoder instance.

NOTE The concept of metadata stream does not yet possess a definition in this document and may be further refined in future editions of this document.

Figure 2 depicts an architecture for handling multiple video decoder instances on a single hardware platform. In this scenario, one or more video decoder instances running on the same video decoder hardware engine are exposed to the application layer as several decoder instances each with their own interface.



Figure 2 — Example relationship between video decoder instances and video decoder hardware engine

## Input video decoding interface

The video decoding engine accepts media streams and metadata streams. There is at least one media stream as input but there is no constraint on the number of metadata streams with respect to the number of media streams being concurrently consumed by the VDE.

The input of the VDE comprises thus:

* n media streams
* m metadata streams

## Output video decoding interface

The video decoding engine outputs decoded video sequences and metadata streams. There is at least one decoded video sequence as output but there is no constraint on the number of metadata streams with respect to the number of decoded video sequences being concurrently output by the VDE.

These two output stream types may be provided in form of multiplexed output buffers, including both decoded media data and its associated metadata.

The output of the VDE comprises thus:

* q decoded sequences
* p metadata streams

## Control interface to the video decoding interface

### Functions

In order to support immersive media applications, Clause 5.4 defines an abstract video decoding interface. A video decoding platform that complies with this document shall implement this video decoding interface whose IDL can be found in Annex A.

The video decoding interface consists of the abstract functions defined in the following subclause. These functions are defined using the IDL syntax specified in ISO/IEC 19516 Information technology — Object management group — Interface definition language (IDL) 4.2.

Figure 3 depicts an example instantiation of decoder instances using some of the functionalities of the video decoding interface. The video decoder instances with identifiers 1 to 3 belong to the group with identifier 4. By this grouping mechanism, the three instances are instructed to write the decoded sequences into a single aggregate buffer and the decoding operations across those instances are performed in a coordinated manner such that no instance runs ahead or behind the others.



Figure 3 — Example instantiation using VDI

#### queryCurrentAggregateCapabilities()

##### Declaration

The IDL declarations of the queryCurrentAggregateCapabilities() function along with the AggregateCapabilities and PerformancePoint structures and the capabilities flags are defined as follows:

const unsigned long CAP\_INSTANCES\_FLAG = 0x1;

const unsigned long CAP\_BUFFER\_MEMORY\_FLAG = 0x2;

const unsigned long CAP\_BITRATE\_FLAG = 0x4;

const unsigned long CAP\_MAX\_SAMPLES\_SECOND\_FLAG = 0x8;

const unsigned long CAP\_MAX\_PERFORMANCE\_POINT\_FLAG = 0xA;

enum ChromaFormat {monochrome=1, YCbCr\_420, YCbCr\_422, YCbCr\_444};

struct PerformancePoint {

float picture\_rate;

unsigned long width;

unsigned long height;

unsigned long bit\_depth;

ChromaFormat chroma\_format;

};

struct AggregateCapabilities {

unsigned long flags;

unsigned long max\_instances;

unsigned long buffer\_memory;

unsigned long bitrate;

unsigned long max\_samples\_second;

PerformancePoint max\_performance\_point;

};

AggregateCapabilities queryCurrentAggregateCapabilities (

in string component\_name,

in unsigned long flags

);

##### Definition

###### General

The queryCurrentAggregateCapabilities() function can be used by the application to query the instantaneous aggregate capabilities of a decoder platform for a specific codec component.

The capability flags can be set separately or in a single function call to query one or more parameters.

###### component\_name

The component\_name provides the name of the component of the decoding platform for which the query applies. The name “All” may be used to indicate that the query is not for a particular component but is rather for all the components of the decoding platform. Components are hardware or software functionalities exposed by the Video Decoding Engine such as decoders.

###### CAP\_INSTANCES\_FLAG and max\_instances

CAP\_INSTANCES\_FLAG queries the max\_instances parameter which indicates the maximum number of decoder instances that can be instantiated at this moment for the provided decoder component.

###### CAP\_BUFFER\_MEMORY\_FLAG and buffer\_memory

CAP\_BUFFER\_MEMORY\_FLAG queries the buffer\_memory parameter which indicates the instantaneous global maximum available buffer size in bytes that can be allocated independently of any components at this moment on the decoder platform for buffer exchange. The allocation of the memory can be done by the application or the VDE itself depending on the VDE instantiation.

###### CAP\_BITRATE\_FLAG and bitrate

CAP\_BITRATE\_FLAG queries the bitrate parameter which indicates the instantaneous maximum coded bitrate in bits per second that the queried component is able to process.

###### CAP\_MAX\_SAMPLES\_SECOND\_FLAG and max\_samples\_second

CAP\_MAX\_SAMPLES\_SECOND\_FLAG queries the max\_samples\_second parameter which indicates the instantaneous maximum number of luma and chroma samples combined per second that the queried component is able to process.

###### CAP\_MAX\_PERFORMANCE\_POINT\_FLAG and max\_performance\_point

CAP\_MAX\_PERFORMANCE\_POINT\_FLAG queries the max\_performance\_point parameter which indicates the maximum performance point of a bitstream that can be decoded by the indicated component in a new instance of that decoder component.

###### PerformancePoint

A performance point contains the following parameters:

* picture\_rate indicating the instantaneous picture rate of the maximum performance point in pictures per second.
* height indicating the height in luma samples of the maximum performance point.
* width indicating the width in luma samples of the maximum performance point.
* bit\_depth indicating the bit depth of the luma samples of the maximum performance point.
* chroma\_format indicating the assumed chroma format for this performance point.

NOTE Each parameter of the max performance point does not necessarily represent the maximum in that dimension. It is the combination of all dimensions that constitutes the maximum performance point.

#### getInstance()

##### Declaration

The IDL declarations of the getInstance() function and the associated ErrorAllocation exception are defined as follows:

exception ErrorAllocation {

string reason;

};

unsigned long getInstance(

in string component\_name,

inout unsigned long group\_id // optional, default value = -1

) raises(ErrorAllocation);

##### Definition

The result of a successful call to the getInstance()function shall provide the identifier of the instance and the group\_id that is assigned or created for this new instance, if one was requested. The default behavior is that the decoder instance does not belong to any already established group but is assigned to a newly created group.

Several decoder instances belonging to the same group means that the VDE treats those instances collectively such that the decoding statuses of those instances progress in synchrony and not in competition against each other. As a consequence, the VDE will also ensure synchronized output writing operations, possibly into an aggregate buffer. There are no conditions for two video decoder instances to be in the same group. In other words, decoder instances may belong to the same group even if their respective components are different.

#### setConfig()

##### Declaration

The IDL declarations of the setConfig() function, the associated ErrorConfig exception, the ConfigDataParameters structure and the ConfigParameters enumeration are defined as follows:

enum ConfigParameters {

CONFIG\_OUTPUT\_BUFFER

};

enum OutputFormat {

OUTPUT\_R;

OUTPUT\_G;

OUTPUT\_B;

OUTPUT\_RGB;

OUTPUT\_RGBA;

OUTPUT\_DEPTH;

OUTPUT\_ALPHA;

OUTPUT\_AUDIO;

};

enum SampleFormat {

SCALAR = 1,

VEC2,

VEC3,

VEC4

};

enum SampleType {

BYTE = 5120,

UNSIGNED\_BYTE,

SHORT,

UNSIGNED\_SHORT,

UNSIGNED\_INT = 5125,

FLOAT

};

struct ConfigDataParameters {

OutputFormat output\_format;

SampleFormat sample\_format;

SampleType sample\_type;

unsigned long sample\_stride;

unsigned long line\_stride;

unsigned long buffer\_offset;

unsigned long output\_buffer\_handle;

};

exception ErrorConfig {

string reason;

};

boolean setConfig (

in unsigned long instance\_id,

in ConfigParameters config\_parameters,

in ConfigDataParameters config\_data\_parameters

) raises(ErrorConfig);

##### Definition

The setConfig() function may be called with the parameter CONFIG\_OUTPUT\_BUFFER, in which case it provides a handle to the output buffer and a description of the write operation into that output buffer.

The parameters that are passed to this function when setting the configuration for CONFIG\_OUTPUT\_BUFFER are as follows:

* sample\_format indicating the format of each sample, which can be a scalar, a 2D vector, a 3D vector, or a 4D vector.
* sample\_type indicating the type of each component of the sample.
* sample\_stride indicating the number of bytes between 2 consecutive samples of this output.
* line\_stride indicating the number of bytes between the first byte of one line and the first byte of the following line of this output.
* buffer\_offset indicating the offset into the output buffer, starting from which the output frame should be written.
* output\_buffer\_handle provides the handle of the output buffer, to which the output of the decoder instance is to be written. The VDE is responsible for the allocation and management of the memory for the output buffer.

#### getParemeter() and setParameter()

##### Declaration

The IDL declarations of the getParameter() and setParemeter() functions as well as the associated ErrorParameter exception and the ExtParameters enumeration are defined as follows:

enum ExtParameters {

PARAM\_PARTIAL\_OUTPUT,

PARAM\_SUBFRAME\_OUTPUT,

PARAM\_METADATA\_CALLBACK,

PARAM\_OUTPUT\_CROP,

PARAM\_MAX\_OFFTIME\_JITTER

};

enum PartialOutput {

PARTIAL\_OUTPUT\_NOT\_ALLOWED = 0;

PARTIAL\_OUTPUT\_ALLOWED;

PARTIAL\_OUTPUT\_DESIRED;

};

struct SubframeOutput {

unsinged int subframe\_id;

};

struct MetadataCallback {

unsigned int metadata\_ids[];

unsigned long metadata\_callback;

};

struct MaxOfftimeJitter {

unsinged int jitter\_millis;

};

struct CropWindow {

unsigned long x;

unsigned long y;

unsigned long width;

unsigned long height;

};

exception ErrorParameter {

string reason;

};

any getParameter (

in unsigned long instance\_id,

in ExtParameters ext\_parameters,

out any parameter

);

boolean setParameter (

in unsigned long instance\_id,

in ExtParameters ext\_parameters,

in any parameter

) raises(ErrorParameter);

##### Definition

###### General

The getParameter() and setParameter() functions can receive the extended parameters in the clauses below.

###### PARAM\_PARTIAL\_OUTPUT

PARAM\_PARTIAL\_OUTPUT indicates whether the output of corrupted/incomplete frames is required, desired, or not allowed. This may be useful to instruct the decoder on how to handle the output of corrupted or incompletely decoded frames as a result of missing or malformed data in the input.

###### PARAM\_SUBFRAME\_OUTPUT

PARAM\_SUBFRAME\_OUTPUT indicates the one or more subframes to be output by the decoder. Subframes may e.g. be auxiliary pictures in a video stream. The identifier of the subframe substream is provided as part of the SubframeOutput structure.

###### PARAM\_METADATA\_CALLBACK

PARAM\_METADATA\_CALLBACK sets a callback function for a specific metadata type. The list of supported metadata types is codec-dependent and shall be defined for each codec independently. The set of metadata types for which the callback is to be invoked by the VDE as well as the callback function handler are provided in the MetadataCallback structure.

###### PARAM\_OUTPUT\_CROP

PARAM\_OUTPUT\_CROP indicates that only part of the decoded frame is desired at the output. The decoder instance may use this information to reduce intelligently its decoding processing by discarding units that do not fall in the cropped output region whenever possible. The information about the area of the video to be output is provided by the CropWindow structure.

###### PARAM\_MAX\_OFFTIME\_JITTER

PARAM\_MAX\_OFFTIME\_JITTER indicates the maximum amount of time in microseconds between consecutive executions of the decoder instance. This parameter is relevant whenever the underlying hardware component is shared among multiple decoder instances, which requires context switching between the different decoder instances. The information about the allowed jitter is provided in the MaxOfftimeJitter structure.