INTERNATIONAL STANDARD

ISO/IEC 23090-2

Second edition
2021-07

Information technology — Coded representation of immersive media —

Part 2: Omnidirectional media format

Technologies de l'information — Représentation codée de média immersifs —

Partie 2: Format de média omnidirectionnel

Reference number
ISO/IEC 23090-2:2021(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

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 document 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 or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC 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) or the IEC list of patent declarations received (see patents.iec.ch).

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. In the IEC, see www.iec.ch/understanding-standards.

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

This second edition cancels and replaces the first edition (ISO/IEC 23090-2:2019), which has been technically revised.

The main changes compared to the previous edition are as follows:

— Multiple viewpoints have been added. Viewpoints can be used for example to provide several user-switchable camera positions to view the content or to express a storyline where the user is given the choice to select which storyline path is followed.

— Sphere-relative and viewport-relative video and image overlays have been added.

— Mesh omnidirectional video where the video is projected on an indicated set of mesh elements has been added.

— Two tiling OMAF video profiles for viewport-dependent streaming have been added.

A list of all parts in the ISO/IEC 23090 series can be found on the ISO and IEC websites.

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 and www.iec.ch/national-committees.

Introduction

When omnidirectional media content is consumed with a head-mounted display and headphones, only the parts of the media that correspond to the user's viewing orientation are rendered, as if the user were in the spot where and when the media was captured. One of the forms of omnidirectional media applications is omnidirectional video, also known as 360° video. Omnidirectional video is typically captured by multiple cameras that cover the entire sphere or at least a large part of the sphere. Compared to traditional media application formats, the end-to-end technology for omnidirectional video (from capture to playback) is more easily fragmented due to various capturing and video projection technologies. From the capture side, there exist many different types of cameras capable of capturing 360° video, and on the playback side there are many different devices that are able to playback 360° video with different processing capabilities. To avoid fragmentation of omnidirectional media content and devices, a standardized format for omnidirectional media applications is specified in this document.

This document defines a media format that enables omnidirectional media applications, focusing on 360° video, images, and audio, as well as associated timed text. What is specified in this document includes (but is not limited to):

- a coordinate system that consists of a unit sphere and three coordinate axes, namely the X (back-to-front) axis, the Y (lateral, side-to-side) axis, and the Z (vertical, up) axis;
- projection and rectangular region-wise packing methods that may be used for conversion of a spherical video sequence or image into a two-dimensional rectangular video sequence or image, respectively;
- storage of omnidirectional media and the associated metadata using the ISO Base Media File Format (ISOBMFF) as specified in ISO/IEC 14496-12;
- storage of video or image overlays and the associated metadata using ISOBMFF;
- encapsulation, signalling, and streaming of omnidirectional media and overlays in a media streaming system, e.g. dynamic adaptive streaming over HTTP (DASH) as specified in ISO/IEC 23009-1 or MPEG media transport (MMT) as specified in ISO/IEC 23008-1;
- media profiles and presentation profiles that provide conformance points for media codecs as well as media coding and encapsulation configurations that may be used for compression, streaming, and playback of the omnidirectional media content;
- toolset brands that provide conformance points for functionalities beyond plain 360° video, images and audio.

This document is organized as follows:

a) Clause 1 specifies the scope of this document.

b) Clause 2 contains the normative references.

c) Clause 3 specifies the terms, definitions, abbreviated terms, arithmetic operations, mathematical functions and other conventions used in this document.

d) Clause 4 contains an overview of this document.

e) Clause 5 specifies a coordinate system used in this document and the equations for the equirectangular and cubemap omnidirectional projection formats, the conversion from the local coordinate axes to the global coordinate axes, and the rectangular region-wise packing.

f) Clause 6 specifies syntax structures that are common for fisheye video and fisheye images.

g) Clause 7 specifies extensions to the ISOBMFF for omnidirectional media as well as for timed metadata for sphere regions. It also specifies generic extensions to ISO/IEC 14496-12 and ISO/IEC 14496-15, which may be used also for other purposes than for omnidirectional media.

h) Clause 8 specifies extensions to DASH for omnidirectional media.

i) Clause 9 specifies extensions to MMT for omnidirectional media.

j) Clause 10 specifies OMAF media profiles.

k) Clause 11 specifies OMAF presentation profiles based on some of the OMAF media profiles specified in Clause 10.

l) Clause 12 specifies OMAF toolset brands.

m) Annex A contains the OMAF DASH XML schema.

n) Annex B specifies the DASH integration of all the OMAF media profiles for timed media specified in Clause 10.

o) Annex C specifies the CMAF integration of some of the OMAF media profiles specified in Clause 10.

p) Annex D describes some schemes for viewport-dependent omnidirectional video processing.

q) Annex E contains some DASH MPD examples.

r) Annex F contains some MMT signalling examples.

s) Annex G specifies the expected OMAF player behaviour for rendering overlays.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this document may involve the use of a patent.

ISO and IEC take no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured ISO and IEC that he/she is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with ISO and IEC. Information may be obtained from the patent database available at www.iso.org/patents.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those in the patent database. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Information technology — Coded representation of immersive media —

Part 2: Omnidirectional media format

1 Scope

This document specifies the omnidirectional media format for coding, storage, delivery and rendering of omnidirectional media, including video, images, audio and timed text. Omnidirectional image or video can contain graphics elements generated by computer graphics but encoded as image or video. Multiple viewpoints, each corresponding to an omnidirectional camera, are supported. The document also specifies storage and delivery of overlay images or video intended to be rendered over the omnidirectional background image or video.

2 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 10918-1, *Information technology — Digital compression and coding of continuous-tone still images — Part 1: Requirements and guidelines*

ISO/IEC 14496-1, *Information technology — Coding of audio-visual objects — Part 1: Systems*

ISO/IEC 14496-3:2019, *Information technology — Coding of audio-visual objects — Part 3: Advanced audio coding*

Rec. ITU-T H.264 (06/19) | ISO/IEC 14496-10:2014, *Information technology — Coding of audio-visual objects — Part 10: Advanced video coding*

ISO/IEC 14496-12:2020, *Information technology — Coding of audio-visual objects — Part 12: ISO base media file format*

ISO/IEC 14496-14, *Information technology — Coding of audio-visual objects — Part 14, MP4 file format*

ISO/IEC 14496-15:2019, *Information technology — Coding of audio-visual objects — Part 15, Carriage of network abstraction layer (NAL) unit structured video in the ISO base media file format*

ISO/IEC 14496-30:2018, *Information technology — Coding of audio-visual objects — Part 30: Timed text and other visual overlays in ISO base media file format*

ISO/IEC 23000-19:2020, *Information technology — Multimedia application format (MPEG-A) — Part 19: Common media application format (CMAF) for segmented media*

ISO/IEC 23000-22:2019, *Information technology — Multimedia application format —Part 22 Multi-image application format (MIAF)*

ISO/IEC 23003-3:2020, *Information technology — MPEG audio technologies — Part 3: Unified speech and audio coding*

ISO/IEC 23003-4:2020, *Information technology — MPEG audio technologies — Part 4: Dynamic range control*

ISO/IEC 23008-1:2017, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 1: MPEG media transport (MMT)*

Rec. ITU-T H.265 | ISO/IEC 23008-2:2020, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 2: High efficiency video coding*

ISO/IEC 23008-3:2019, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 3: 3D audio*

ISO/IEC 23008-12, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 12: Image file format*

ISO/IEC 23009-1:2019, *Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats*

ISO/IEC 23091-2, *Information technology — Coding-independent code points — Part 2: Video*

ISO/IEC 23091-3, *Information technology — Coding-independent code points — Part 3: Audio*

IETF BCP 47, *Tags for Identifying Languages*

IETF Internet Standard 66, *Uniform Resource Identifier (URI): Generic Syntax*

IETF RFC 6381, *MIME Codecs and Profiles*

W3C Candidate Recommendation, *WebVTT, The Web Video Text Tracks Format*

W3C Recommendation, *TTML Profiles for Internet Media Subtitles and Captions 1.0.1 (IMSC1)*

W3C Recommendation, *XML schema part 1: Structures*

W3C Recommendation, *XML schema part 2: Datatypes*

W3C Recommendation, *XML Path Language (XPath) 2.0 (Second Edition)*

3 Terms, definitions, abbreviated terms and conventions

3.1 Terms and definitions

For the purposes of this document, the terms and definitions in ISO/IEC 14496-12, ISO/IEC 23008-12 and the following apply.

NOTE In particular, the terms coded image, coded image item, derived image, derived image item, image item, reconstructed image and source image item are defined in ISO/IEC 23008-12.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1.1
azimuth

first of the two sphere coordinates describing the location of a point on the sphere

Note 1 to entry: Azimuth and elevation are specified in subclause 5.1.

3.1.2
azimuth circle

circle on the sphere connecting all points with the same azimuth value

Note 1 to entry: An *azimuth circle* is always a *great circle* (3.1.22).

3.1.3
background visual media

piece of *visual media* (3.1.62) on which an *overlay* (3.1.31) is superimposed

3.1.4
circular image

image captured with a *fisheye lens* (3.1.18)

3.1.5
closed scheme type

scheme type (3.1.46) that clearly specifies which transformations are allowed and does not allow future extensions

3.1.6
common reference coordinate system

3D Cartesian coordinate system with the centre being (X, Y, Z) equal to (0, 0, 0), used as the reference coordinate system for all *viewpoints* (3.1.59) within a *viewpoint group* (3.1.60)

3.1.7
composition-aligned sample

sample in a track that is associated with another track, the sample has the same composition time as a particular sample in the another track, or, when a sample with the same composition time is not available in the another track, the closest preceding composition time relative to that of a particular sample in the another track

3.1.8
composition picture

picture that is suitable to be presented and is obtained from the decoding outputs of *composition-aligned samples* (3.1.7) of all tracks of a 2D spatial relationship track group by arranging them spatially as specified by the semantics of the 2D spatial relationship track group

3.1.9
constituent picture

such part of a spatially frame-packed stereoscopic picture that corresponds to one view, or a picture itself when frame packing is not in use or the temporal interleaving frame packing arrangement is in use

3.1.10
content coverage

one or more *sphere regions* (3.1.48) that are covered by the content represented by the track, an image item, or a *composition picture* (3.1.8)

3.1.11
elevation

second of the two sphere coordinates describing the location of a point on the sphere

Note 1 to entry: Azimuth and elevation are specified in subclause 5.1.

3.1.12
elevation circle

circle on the sphere connecting all points with the same elevation value

Note 1 to entry: When the elevation is zero, an *elevation circle* is also a *great circle* (3.1.22). This coincides with the equator on Earth.

3.1.13
encoded tile sequence

coded representation of a *tile sequence* (3.1.52) that has the capability to be merged with other encoded tile sequences in coded domain without decoding mismatch by rewriting only header data

3.1.14
extractor track

track that has untransformed sample entry type equal to `'hvc2'`, `'avc2'`, or `'avc4'` and contains one or more `'scal'` track references

3.1.15
field of view

extent of the observable world in captured/recorded content or in a physical display device

3.1.16
file decoder

collective term for file/segment decapsulation and decoding of video, audio or image bitstreams

3.1.17
file decoding process

process specified as a part of a media profile specification that takes as input a set of ISOBMFF tracks or items and derives either a decoded pictures or audio samples, and rendering metadata for them; or a fully rendered audio scene in the reference system

3.1.18
fisheye lens

wide-angle camera lens that usually captures an approximately hemispherical *field of view* (3.1.15) and projects it as a *circular image* (3.1.4)

3.1.19
fisheye omnidirectional video
fisheye omnidirectional image

omnidirectional media (3.1.29) where *circular images* (3.1.4) of *fisheye lenses* (3.1.18) are spatially arranged onto picture(s)

3.1.20
fisheye video

video captured by *fisheye lenses* (3.1.18)

3.1.21
global coordinate axes

coordinate axes that are associated with audio, video, and images representing the same acquisition position and intended to be rendered together

Note 1 to entry: Coordinate axes are specified in subclause 5.1.

Note 2 to entry: The origin of the global coordinate axes is usually the same as the centre point of a device or rig used for omnidirectional audio/video acquisition as well as the position of the observer's head in the three-dimensional space in which the audio and video tracks are located.

Note 3 to entry: In the absence of the initial viewing orientation metadata (see subclause 7.7.4 for tracks or subclause 7.9.9 for image items), the initial viewing orientation should be inferred to be equal to (0, 0, 0) for (`centre_azimuth`, `centre_elevation`, `centre_tilt`) relative to the global coordinate axes.

3.1.22
great circle

intersection of the sphere and a plane that passes through the centre point of the sphere

Note 1 to entry: A *great circle* is also known as an orthodrome or Riemannian circle.

Note 2 to entry: The centre of the sphere and the centre of a *great circle* are co-located.

3.1.23
guard band

area that is not rendered but may be used to improve the rendering quality to avoid or mitigate visual artifacts such as seams

Note 1 to entry: Guard bands in *packed pictures* (3.1.34) are associated with *packed regions* (3.1.35) as described in subclause 7.5.3.

3.1.24
local coordinate axes

coordinate axes obtained after applying rotation to the *global coordinate axes* (3.1.21)

3.1.25
mesh omnidirectional video

omnidirectional media (3.1.29) where rectangular regions of two-dimensional pictures are mapped to mesh elements of a three-dimensional mesh

3.1.26
OMAF base track

track whose samples are resolved by merging samples of the referenced *OMAF tile tracks* (3.1.28)

Note 1 to entry: An HEVC tile base track as specified in ISO/IEC 14496-15 and an *extractor track* (3.1.14) referencing OMAF tile tracks are examples of OMAF base tracks.

3.1.27
OMAF player

application responsible for receiving files or segments or accessing files locally, decapsulating these files and segments, decoding the audio, video, image or timed text bitstreams, rendering the audio, video, image or timed text and applying a strategy for receiving the files or segments based on the *viewport* (3.1.61) information

3.1.28
OMAF tile track

track that represents a rectangular subset of the original video content and has the capability to be merged with other OMAF tile tracks in coded domain without decoding mismatch by rewriting only header data, where a decoding mismatch refers to the value of any pixel having a different value when decoded prior to merging or subsequently to merging

Note 1 to entry: An HEVC tile track as specified in ISO/IEC 14496-15 is an example of an OMAF tile track.

3.1.29
omnidirectional media

media such as image or video and its associated audio that enable rendering according to the user's *viewing orientation* (3.1.56), if consumed with a head-mounted device, or according to user's desired *viewport* (3.1.61), otherwise, as if the user was in the spot where and when the media was captured

3.1.30
open-ended scheme type

scheme type (3.1.46) that allows future extensions

3.1.31
overlay

piece of *visual media* (3.1.62) rendered over omnidirectional video or image item or over a *viewport* (3.1.61)

3.1.32
overlay source

piece of *visual media* (3.1.62) used as the content for an *overlay* (3.1.31)

3.1.33
overlay source region

overlay source (3.1.32) specified as a rectangular region within decoded pictures

3.1.34
packed picture

picture that is represented as a coded picture in the coded video bitstream

Note 1 to entry: A *packed picture* may result from *region-wise packing* (3.1.43) of a *projected picture* (3.1.37).

3.1.35
packed region

region in a *packed picture* (3.1.34) that is mapped to a *projected region* (3.1.38) as specified by the region-wise packing signalling

3.1.36
projected omnidirectional video
projected omnidirectional image

omnidirectional media (3.1.29) where the relation of two-dimensional picture coordinates to *sphere coordinates* (3.1.47) is specified using mathematical projection equation(s)

3.1.37
projected picture

picture that has a representation format specified by an omnidirectional video projection format

Note 1 to entry: Omnidirectional projection formats are specified in subclause 5.2.

Note 2 to entry: A project picture i) covers the entire sphere. ii) has no re-sized, re-positioned, rotated, or mirrored regions relative to a representation format specified by an omnidirectional video projection format, and iii) has no *guard bands* (3.1.23).

3.1.38
projected region

region in a *projected picture* (3.1.37) that is mapped to a *packed region* (3.1.35) as specified by the region-wise packing signalling

3.1.39
projection

inverse of the process by which the samples of a *projected picture* (3.1.37) are mapped to a set of positions identified by a set of *azimuth* (3.1.1) and *elevation* (3.1.11) coordinates on a unit sphere

3.1.40
quality ranking region

region that is associated with a quality ranking value and is specified relative to a decoded picture or a sphere

3.1.41
quality ranking 2D region

quality ranking region (3.1.40) that is specified relative to a decoded picture

3.1.42
quality ranking sphere region

quality ranking region (3.1.40) that is specified relative to a sphere

3.1.43
region-wise packing

inverse of the process of transformation, resizing, and relocating of *packed regions* (3.1.35) of a *packed picture* (3.1.34) to remap to *projected regions* (3.1.38) of a *projected picture* (3.1.37)

3.1.44
rendering

process of generating audio-visual content for playback from the decoded audio-visual data according to the user's *viewing orientation* (3.1.56), if consumed with a head-mounted device, or according to user's desired *viewport* (3.1.61), otherwise, as well as optionally the user's *viewing position* (3.1.57) when sphere-relative *overlays* (3.1.31) are displayed with *background visual media* (3.1.3)

3.1.45
sample

all the data associated with a single time or single element in one of the three sample arrays that represent a picture

Note 1 to entry: When the term sample is used in the context of a track, it refers to all the data associated with a single time of that track, where a time is either a decoding time or a composition time. When the term sample is used in the context of a picture, e.g. in the phrase "luma sample", it refers to a single element in one of the three sample arrays that represent the picture.

3.1.46
scheme type

type that parameterizes an encrypted, restricted, or incomplete media track

3.1.47
sphere coordinates

azimuth (3.1.1) and *elevation* (3.1.11) that identify a location of a point on the unit sphere

3.1.48
sphere region

region on a sphere, specified either by four *great circles* (3.1.22) or by two *azimuth circles* (3.1.2) and two *elevation circles* (3.1.12), or such a region on the rotated sphere after applying certain amount of yaw, pitch, and roll rotations

3.1.49
sub-picture

picture that represents a spatial subset of the original video content, which has been split into spatial subsets before video encoding at the content production side

3.1.50
sub-picture bitstream

bitstream that represents a spatial subset of the original video content, which has been split into spatial subsets before video encoding at the content production side

3.1.51
sub-picture track

track that is with spatial relationships to other track(s) and that represents a *sub-picture bitstream* (3.1.50)

3.1.52
tile sequence

rectangular subset of the original video content

3.1.53
tilt angle

angle indicating the amount of tilt of a *sphere region* (3.1.48), measured as the amount of rotation of the *sphere region* along the axis originating from the sphere origin passing through the centre point of the *sphere region*, where the angle value increases clockwise when looking from the origin towards the positive end of the axis

3.1.54
time-parallel sample

sample in a track that is associated with another track, and either has the same decoding time as a particular sample in the other track, or, when a sample with the same decoding time in the other track is not available, the closest preceding decoding time relative to that of a particular sample in the other track

3.1.55
track sample entry type

sample entry type of a `SampleEntry` directly contained in `SampleDescriptionBox`

3.1.56
viewing orientation

triplet of *azimuth* (3.1.1), *elevation* (3.1.11), and *tilt angle* (3.1.53) characterizing the orientation that a user is consuming the audio-visual content; in case of image or video, characterizing the orientation of the *viewport* (3.1.61)

3.1.57
viewing position

position from which video and image content is viewed

Note 1 to entry: In the first edition of this document, the viewing position is the centre of the unit sphere. In Edition 2 of this document, a viewing position within the *viewing space* (3.1.58) is enabled to view *overlays* (3.1.31) from different perspectives relative to the *background visual media* (3.1.3). When the content is rendered with a head-mounted display, head tracking can be used to locate the viewing position.

3.1.58
viewing space

three-dimensional space of *viewing positions* (3.1.57) within which *rendering* (3.1.44) of image and video is enabled and VR experience is valid

3.1.59
viewpoint

omnidirectional media (3.1.29) corresponding to one omnidirectional camera

3.1.60
viewpoint group

group of *viewpoints* (3.1.59) that share the same *common reference coordinate system* (3.1.6)

3.1.61
viewport

region of *omnidirectional media* (3.1.29) suitable for display and viewing by the user

3.1.62
visual media

video, image item or timed text

3.2 Abbreviated terms

2D	two-dimensional
CICP	coding-independent code points (specified in ISO/IEC 23091-1, 23091-2, and 23091-3)
CMAF	common media application format (specified in ISO/IEC 23000-19)
CMP	cubemap projection
DASH	MPEG dynamic adaptive streaming over HTTP (specified in ISO/IEC 23009-1)
ERP	equirectangular projection
FOV	field of view
GPS	global positioning system
ID	identifier
ISOBMFF	ISO base media file format (specified in ISO/IEC 14496-12)
HEVC	high efficiency video coding (specified in Rec. ITU-T H.265 | ISO/IEC 23008-2)
HMD	head-mounted display
HOA	higher-order ambisonics
MCTS	motion-constrained tile set
MMT	MPEG media transport (specified in ISO/IEC 23008-1)
OMAF	omnidirectional media format (specified in ISO/IEC 23090-2)
SRD	spatial relationship description
URN	uniform resource name
VR	virtual reality
WGS	world geodetic system

3.3 Conventions

3.3.1 Arithmetic operators and mathematical functions

+	addition
−	subtraction (as a two-argument operator) or negation (as a unary prefix operator)

*	multiplication, including matrix multiplication
x^y	exponentiation Specifies x to the power of y. In other contexts, such notation is used for superscripting not intended for interpretation as exponentiation.
/	integer division with truncation of the result toward zero For example, 7 / 4 and −7 / −4 are truncated to 1 and −7 / 4 and 7 / −4 are truncated to −1.
÷	division in mathematical equations where no truncation or rounding is intended
$\frac{x}{y}$	division in mathematical equations where no truncation or rounding is intended
$\sum_{i=x}^{y} f(i)$	summation of f(i) with i taking all integer values from x up to and including y
x % y	modulus Remainder of x divided by y, defined only for integers x and y with x >= 0 and y > 0.

$$\text{Abs}(\,x\,) = \begin{cases} x & ; \quad x >= 0 \\ -x & ; \quad x < 0 \end{cases} \tag{1}$$

Asin(x) trigonometric inverse sine function, operating on an argument x that is in the range of −1.0 to 1.0, inclusive, with an output value in the range of −π÷2 to π÷2, inclusive, in units of radians

Atan(x) trigonometric inverse tangent function, operating on an argument x that is any real number, with an output value in the range of −π÷2 to π÷2, inclusive, in units of radians

$$\text{Atan2}(\,y, x\,) = \begin{cases} \text{Atan}\left(\frac{y}{x}\right) & ; \quad \text{if } x > 0 \\ \text{Atan}\left(\frac{y}{x}\right) + \pi & ; \quad \text{if } x < 0 \ \&\& \ y >= 0 \\ \text{Atan}\left(\frac{y}{x}\right) - \pi & ; \quad \text{if } x < 0 \ \&\& \ y < 0 \\ +\frac{\pi}{2} & ; \quad \text{if } x == 0 \ \&\& \ y >= 0 \\ -\frac{\pi}{2} & ; \quad \text{otherwise} \end{cases} \tag{2}$$

Cos(x) trigonometric cosine function operating on an argument x in units of radians

Floor(x) largest integer less than or equal to x

$$\text{Round}(\,x\,) = \text{Sign}(\,x\,) * \text{Floor}(\,\text{Abs}(\,x\,) + 0.5\,) \tag{3}$$

$$\text{Sign}(\,x\,) = \begin{cases} 1 & ; \quad x > 0 \\ 0 & ; \quad x == 0 \\ -1 & ; \quad x < 0 \end{cases} \tag{4}$$

Sin(x) trigonometric sine function operating on an argument x in units of radians

Tan(x) trigonometric tangent function operating on an argument x in units of radians

3.3.2 Order of operation precedence

The following notation is used to specify a range of values:

When order of precedence in an expression is not indicated explicitly by use of parentheses, the following rules apply:

— Operations of a higher precedence are evaluated before any operation of a lower precedence.

— Operations of the same precedence are evaluated sequentially from left to right.

Table 1 specifies the precedence of operations from highest to lowest; a higher position in the table indicates a higher precedence.

NOTE For those operators that are also used in the C programming language, the order of precedence used in this document is the same as used in the C programming language.

Table 1 — Operation precedence from highest (at top of table) to lowest (at bottom of table)

operations (with operands x, y, and z)
"x++", "x− −"
"!x", "−x" (as a unary prefix operator)
x^y
"x * y", "x / y", "x ÷ y", "$\frac{x}{y}$", "x % y"
"x + y", "x − y" (as a two-argument operator), "$\sum_{i=x}^{y} f(i)$"
"x << y", "x >> y"
"x < y", "x <= y", "x > y", "x >= y"
"x = = y", "x != y"
"x & y"
"x | y"
"x && y"
"x || y"
"x ? y : z"
"x..y"
"x = y", "x += y", "x −= y"

3.3.3 Range notation

The following notation is used to specify a range of values:

x = y..z	x takes on integer values starting from y to z, inclusive, with x, y, and z being integer numbers and z being greater than y.

3.3.4 Variables

This document derives variables that are named by a mixture of lower case and upper case letter and without any underscore characters. Variables starting with an upper case letter are derived for the current syntax structure and all depending syntax structures. Variables starting with an upper case letter may be used in the specification for dependent syntax structures without mentioning the originating syntax structure of the variable. Variables starting with a lower case letter are only used within the subclause in which they are derived.

3.3.5 Processes

Processes are used to describe the decoding of syntax elements. A process has a separate specification and invoking. All syntax elements and upper case variables that pertain to the current syntax structure and depending syntax structures are available in the process specification and invoking. A process specification may also have a lower case variable explicitly specified as input. Each process specification has explicitly specified an output. The output is a variable that can either be an upper case variable or a lower case variable.

When invoking a process, the assignment of variables is specified as follows:

— If the variables at the invoking and the process specification do not have the same name, the variables are explicitly assigned to lower case input or output variables of the process specification.

— Otherwise (the variables at the invoking and the process specification have the same name), assignment is implied.

3.3.6 Conventions for indicating the number of boxes in tables

This subclause specifies conventions used in box tables (i.e. Table 30, Table 31, Table 32, Table 33, Table 34, Table 50, Table 51, Table 52, Table 53, Table 54, Table 55, Table B.1, Table B.2, and Table B.3) where the nesting and the number of boxes with given four-character codes are specified.

The Format Req. column in the tables specifies the number of boxes that shall be present as follows:

— "*" specifies that "zero or more" boxes may be present.

— "+" specifies that "one or more" boxes shall be present.

— "0/1" specifies that either zero or one boxes shall be present.

— "1" specifies that one and only one box shall be present.

In addition to boxes included in a box table, any other boxes may be present unless specifically disallowed.

4 Overview

4.1 Organization of this clause

This clause is organized as follows:

— Subclause 4.2 provides the overall architecture for an omnidirectional media application.

— Subclauses 4.3, 4.4, and 4.5 specify how the overall architecture is applied for projected, fisheye, and mesh omnidirectional video/images, respectively.

— Subclause 4.6 describes applicable streaming methods for video delivery.

— Subclause 4.7 describes which types of referenceable features specified in this document implementations and external specifications may be used to achieve interoperability. It also defines OMAF media and presentation profiles.

4.2 Overall architecture

Figure 1 shows a typical content flow process for an omnidirectional media application.

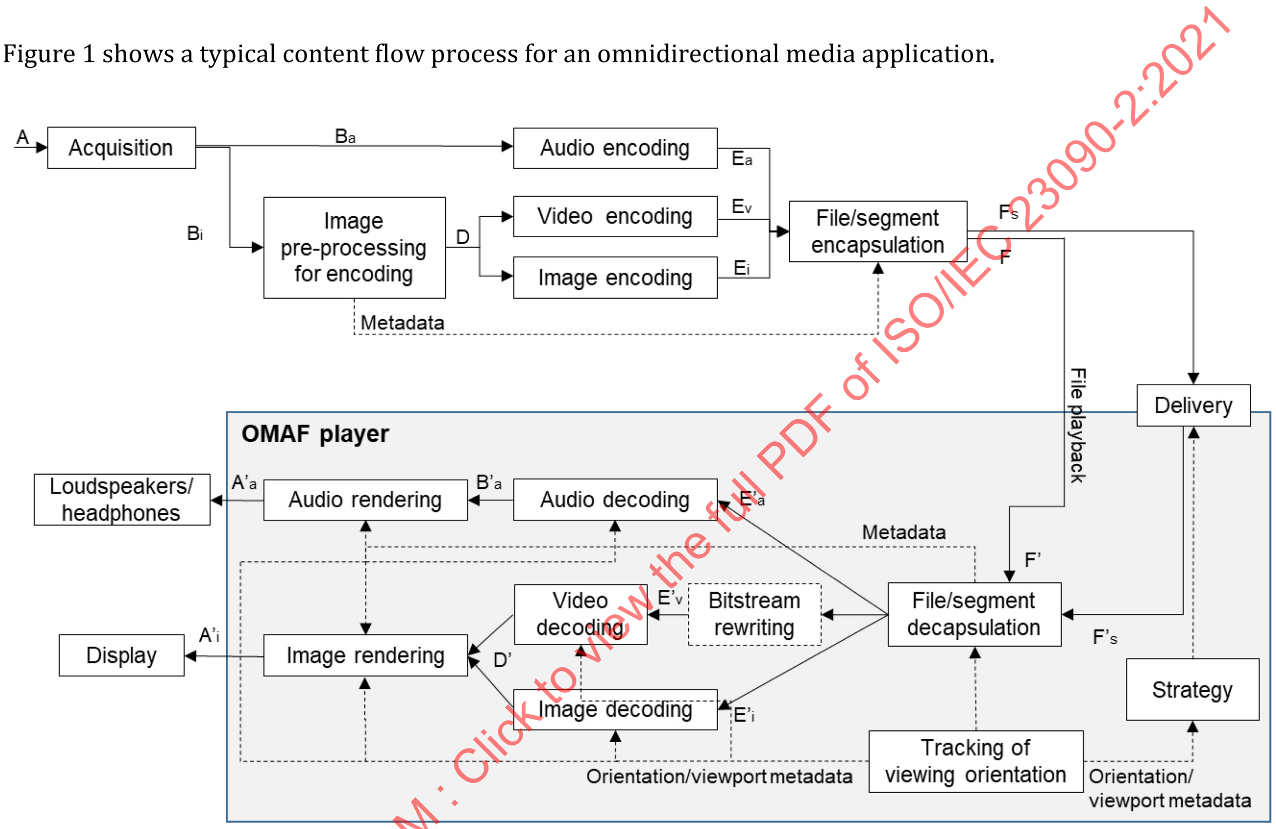


Figure 1 — Content flow process for omnidirectional media with projected video

The following interfaces are specified in this document:

— E'_a, E'_v, E'_i: audio bitstream, video bitstream, coded image(s), respectively; see Clause 10.

— F/F': media file; see Clause 7. Moreover, media profiles specified in Clause 10 include the specification of the track formats for F/F', which may contain constraints on the elementary streams contained within the samples of the tracks.

— Clause 8 specifies the delivery related interfaces for DASH delivery.

— Clause 9 specifies the delivery related interfaces for MMT delivery.

The other interfaces in Figure 1 are not specified in this document.

NOTE While the syntax and semantics of the bitstreams E_a, E_v, and E_i are the same as those for E'_a, E'_v, E'_i, respectively, the input interface to the file/segment encapsulation module is not specified.

A real-world audio-visual scene (A) is captured by audio sensors as well as a set of cameras or a camera device with multiple lenses and sensors. The acquisition results in a set of digital image/video (B_i) and audio (B_a) signals. The cameras/lenses typically cover all directions around the centre point of the camera set or camera device, thus the name of 360-degree video.

Audio may be captured using many different microphone configurations and stored as several different content formats, including channel-based signals, static or dynamic (i.e. moving through the 3D scene) object signals, and scene-based signals (e.g. Higher Order Ambisonics). The channel-based signals typically conform to one of the loudspeaker layouts defined in ISO/IEC 23091-3. In an omnidirectional media application, the loudspeaker layout signals of the rendered immersive audio program are binauralized for presentation via headphones.

For audio, no stitching process is needed, since the captured signals are inherently immersive and omnidirectional.

This document specifies the following types of omnidirectional video and images, which differ in the architecture in the image pre-preprocessing for encoding and in the image rendering processing blocks:

- Projected omnidirectional video/images:
  - Image pre-processing for encoding: The images (B_i) of the same time instance are stitched, possibly rotated, and projected onto a 2D picture coordinates using a mathematically specified projection format. Optionally, the resulting projected pictures may be further mapped region-wise onto a packed picture. Either projected pictures or packed pictures are subject to video or image encoding.
  - Image rendering: Either regions of the decoded packed pictures (if region-wise packing has been applied) or the entire projected picture (otherwise) is mapped onto a rendering mesh suitable for the projection format in use.
- Fisheye omnidirectional video/images:
  - Image pre-processing for encoding: Circular images (Bi) captured by fisheye lenses are arranged onto a 2D picture, which is then input to video or image encoding.
  - Image rendering: The decoded circular images are stitched using the signalled fisheye-specific parameters.
- Mesh omnidirectional video:
  - Image pre-processing for encoding: A 3D mesh consisting of mesh elements is generated, where mesh elements can be either parallelograms or regions of a sphere surface. The images (Bi) of the same time instance are stitched, possibly rotated, and projected onto the 3D mesh. Mesh elements are mapped onto rectangular regions of one or more 2D pictures, which are input to video encoding.
  - Image rendering: Rectangular regions of the decoded 2D picture(s) are mapped to the 3D mesh, which is used directly as the rendering mesh.

Further details of the architecture for projected, fisheye, and mesh omnidirectional video/images are provided in subclauses 4.3, 4.4, and 4.5, respectively.

The pre-processed pictures (D) are encoded as coded images (E_i) or a coded video bitstream (E_v). The captured audio (B_a) is encoded as an audio bitstream (E_a). The coded images, video or audio are then composed into a media file for file playback (F) or a sequence of an initialization segment and media segments for streaming (F_s), according to a particular media container file format. In this document, the media container file format is the ISO Base Media File Format specified in ISO/IEC 14496-12. The file encapsulator also includes metadata into the file or the segments.

The segments F_s are delivered using a delivery mechanism to a player.

The file that the file encapsulator outputs (F) is identical to the file that the file decapsulator inputs (F'). A file decapsulator processes the file (F') or the received segments (F'_s) and extracts the coded bitstreams (E'_a, E'_v, or E'_i) and parses the metadata. Viewport-dependent video may be carried in multiple tracks, which may be merged in the bitstream rewriting into a single video bitstream E'_v prior to decoding. The audio, video or images are then decoded into decoded signals (B'_a for audio, and D' for images/video). In the image rendering block, the decoded pictures (D') are projected onto the screen of a head-mounted display or any other display device based on the metadata parsed from the file. Likewise, decoded audio (B'_a) is rendered, e.g. through headphones, according to the current viewing orientation. The current viewing orientation is determined by the viewing orientation tracking functionality. When a head-mounted display is in use, the viewing orientation tracking can involve head tracking and possibly also eye tracking. When sphere-relative overlays are in use, the viewing orientation tracking functionality can include or be complemented by viewing position tracking and rendering of overlays with background visual media can take both the viewing position and the viewing orientation into account. Besides being used by the renderer to render the appropriate part of decoded video and audio signals, the current viewing orientation may also be used by the video and audio decoders for decoding optimization. In viewport-dependent delivery, the current viewing orientation is also passed to the strategy module, which determines the video tracks to be received based on the viewing orientation.

The process described above is applicable to both live and on-demand use cases.

4.3 Projected omnidirectional video/images

4.3.1 General

The architecture specified in subclause 4.2 applies with the following further specifications.

The images (B_i) of the same time instance are stitched, possibly rotated, projected, and mapped onto a packed picture (D). A description of this process is provided in subclause 4.3.2.

The packed pictures (D) are encoded as coded images (E_i) or a coded video bitstream (E_v).

The metadata in the file includes information for the following:

— the projection format of the projected picture;

— the area of the spherical surface covered by the packed picture;

— the rotation for converting between the global coordinate axes and the local coordinate axes;

— region-wise packing;

— region-wise quality ranking.

In the image rendering block, the decoded packed pictures (D') are projected onto the screen of a head-mounted display or any other display device based on the current viewing orientation or viewport and the projection, spherical coverage, rotation, and region-wise packing metadata parsed from the file.

4.3.2 Stitching, rotation, projection, and region-wise packing

For monoscopic 360-degree video, the source images of one time instance are stitched, possibly rotated, and projected to generate a projected picture representing one view. The breakdown of the image stitching, rotation, projection, and region-wise packing processes for monoscopic content is illustrated with Figure 2 and described as follows. Source images (B_i) are stitched, possibly rotated, and projected onto a unit sphere.

The image data on the unit sphere is further arranged onto a two-dimensional projected picture (C). The projected picture covers the entire sphere. Optionally, region-wise packing is then applied to map the projected picture onto a packed picture. If the region-wise packing is not applied, the packed picture is identical to the projected picture, and this picture is given as input to image/video encoding. Otherwise, projected regions are mapped onto a packed picture (D) by indicating the location, shape, and size of each packed region in the packed picture, and the packed picture (D) is given as input to image/video encoding. The packed picture may cover only a part of the entire sphere. In practice, the source images may be converted to a packed picture in one process without intermediate steps.

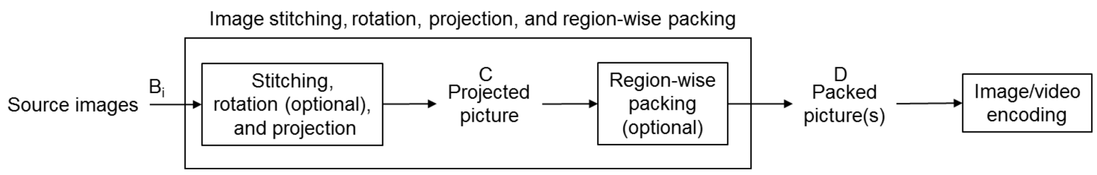


Image stitching, rotation, projection, and region-wise packing
Source images
B_i
Stitching, rotation (optional), and projection
C Projected picture
Region-wise packing (optional)
D Packed picture(s)
Image/video encoding

Figure 2 — Monoscopic image stitching, rotation, projection, and region-wise packing of source images of a single acquisition time instance

In the case of stereoscopic 360-degree video, the source images of one time instance are stitched, possibly rotated, and projected to generate a projected picture representing two views, one for each eye. Both views may be mapped onto the same packed picture, as described below in relation to Figure 3, and encoded by a traditional 2D video encoder. Alternatively, each view of the projected picture may be mapped to its own packed picture, in which case the image stitching, projection, and region-wise packing is like described above with Figure 2. A sequence of packed pictures of either the left view or the right view may be independently coded. Alternatively, packed pictures of the left view and the right view may be temporally interleaved in an alternating manner and encoded as a single bitstream, which is indicated to comply with the temporal interleaving frame packing arrangement. In another alternative, multiview coding is used to encode both views into the same bitstream with inter-view prediction.

The breakdown of the image stitching, rotation, projection, and region-wise packing processes for stereoscopic content where both views are mapped onto the same packed picture is illustrated with Figure 3 and described as follows. Source images (B_i) are stitched, possibly rotated, and projected onto two unit spheres, one for each eye. The image data on each unit sphere is further arranged onto a two-dimensional projected picture (C_L for left eye, C_R for right eye), which covers the entire sphere. Frame packing is applied to pack the left view picture and right view picture onto the same projected picture. Optionally, region-wise packing is then applied to pack the projected picture onto a packed picture, and the packed picture (D) is given as input to image/video encoding. If the region-wise packing is not applied, the packed picture is identical to the projected picture, and this picture is given as input to image/video encoding. The packed picture may cover only a part of the entire sphere.

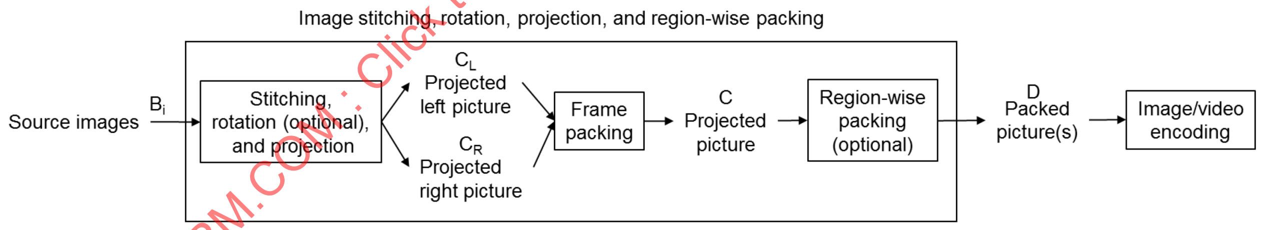


Image stitching, rotation, projection, and region-wise packing
Source images
B_i
Stitching, rotation (optional), and projection
C_L Projected left picture
C_R Projected right picture
Frame packing
C Projected picture
Region-wise packing (optional)
D Packed picture(s)
Image/video encoding

Figure 3 — Stereoscopic image stitching, rotation, projection, and region-wise packing of source images of a single acquisition time instance

The image stitching, rotation, projection, and region-wise packing processes may be carried out multiple times for the same source images to create different versions of the same content, e.g. for different sphere rotations (i.e. different rotations of local coordinate axes with respect to the global coordinate axes). Similarly, the region-wise packing process may be performed multiple times from the same projected picture to create more than one sequence of packed pictures to be encoded.

4.4 Fisheye omnidirectional video/images

The architecture specified in subclause 4.2 applies with the following further specifications.

A fisheye lens is a lens that achieves extremely wide angles of view and produces a circular image. Circular images captured by multiple fisheye lenses of a 360 camera typically cover all directions around the centre point of the camera set or camera device.

The circular images (B_i) captured by fisheye lenses at the same time instance are not stitched, but directly mapped onto a picture (D) in the fisheye video format, which is specified in Clause 6. Region-wise packing is not allowed for the fisheye video format.

The fisheye images including non-stitched circular images (D) are encoded as coded images (E_i) or a coded video bitstream (E_v). The coded images, video or audio are then composed into a media file for file playback (F) or a sequence of an initialization segment and media segments for streaming (F_s), according to a particular media container file format. The file encapsulator also includes fisheye-specific metadata, which assist in rendering the decoded pictures, into the file or the segments.

The fisheye-specific metadata in the file includes:

— Lens distortion correction (LDC) parameters with local variation of FOV,

— Lens shading compensation (LSC) parameters with RGB gains,

— Displayed field of view information, and

— Camera extrinsic parameters (physical location).

In an OMAF player, the circular images in the decoded fisheye video are stitched using the signalled fisheye-specific parameters to an omnidirectional image and rendered according to the user's viewport.

4.5 Mesh omnidirectional video

The architecture specified in subclause 4.2 applies with the following further specifications.

The images (B_i) of the same time instance are stitched, possibly rotated, projected onto a 3D mesh. A picture (D) to be encoded consists of rectangular regions, each mapping to exactly one mesh element of the 3D mesh.

The metadata in the file includes information for the following:

— the rotation for converting between the global coordinate axes and the local coordinate axes,

— the 3D mesh, consisting of mesh elements,

— a mapping of rectangular regions to mesh elements.

The decoded pictures (D') are mapped onto the 3D mesh used for rendering on a head-mounted display or any other display device based on the current viewing orientation or viewport.

4.6 Streaming methods for omnidirectional video

4.6.1 Overview

In viewport-independent streaming, omnidirectional video is represented with the equirectangular projection or another projection method. The video pictures are encoded as a single-layer bitstream using temporal inter prediction, the entire coded bitstream is stored at a server, and if needed, typically fully transmitted to the OMAF player, fully decoded by the decoder, and the area of the decoded picture corresponding to the current viewport is rendered to the user.

A distinct feature of omnidirectional video is that only a viewport is displayed at any particular time, while in normal video applications typically the entire video is displayed. This feature may be utilized to improve the performance of omnidirectional video systems, through selective delivery depending on the user's viewport (or any other criteria, such as recommended viewport timed metadata). Viewport-dependent delivery can be enabled for example by region-wise packing or viewport-dependent video coding. The performance improvement can be either or both of lower transmission bandwidth and lower decoding complexity compared to conventional omnidirectional video systems under the same resolution/quality of the video part presented to the user.

Viewport-dependent delivery can be achieved by either of the following:

— Multiple versions of the content, where different versions are optimized for different viewing orientations. Examples of this category are provided in Clauses D.2 and D.9.

— Splitting the content into tile sequences. Each tile sequence can be encoded at multiple resolutions or at different quality levels. Viewport-dependent delivery can be achieved by receiving different sets of available encoded tile sequences.

The tile-based viewport-dependent streaming can be arranged in one of the following ways:

a) Author-driven binding, which is further categorized into the following two methods:

   1) Viewport-specific author-driven binding: The content author prepares multiple OMAF base tracks, each targeted at a particular range of viewing orientations and providing bitstream rewriting instructions to merge specific encoded tile sequences. The strategy module selects between available OMAF base tracks as described in further detail in subclause 4.6.2. The bitstream rewriting module follows the instructions of the selected OMAF base track to create a video bitstream. Examples of this category are provided in Clause D.6 (excluding subclause D.6.6) and Clause D.8.

   2) Free-viewport author-driven binding. The content author prepares one or more OMAF base tracks. An OMAF base track is not tailor-made for any viewing orientation but gives multiple options from which the OMAF player can choose the encoded tile sequences that are used for extraction. Furthermore, an OMAF base track provides bitstream rewriting instructions to merge any encoded tile sequences selected by the OMAF player. The strategy module selects an OMAF base track and the encoded tile sequences among the indicated options. For example, an OMAF base track can enable the player to choose encoded tile sequences from multiple resolutions and the player can e.g. receive a full set of low-resolution tiles to show the full representation of the content at a lower quality while retrieving a subset of high-resolution tiles that together make up the viewport. The operation of the strategy module is described in subclause 4.6.3. The bitstream rewriting module follows the instructions of the selected OMAF base track to create a video bitstream. Examples of this category are provided in Clause D.4 and subclause D.6.6.

b) Late binding. The strategy module selects any encoded tile sequences that can be merged to a bitstream. It will typically retrieve a full set of low-resolution tiles to show the full representation of the content at a lower quality while retrieving a subset of high-resolution tiles that together make up the viewport. The tile selection strategy can be tailored according to the current viewport, available bandwidth, the field of view of the head-mounted display or the hardware decoding capabilities, among other things. The bitstream rewriting module merges the selected encoded tile sequences into a bitstream. The decoded picture/texture contains a combination of high and low resolution tiles which are then mapped onto a sphere, where the low quality tiles are used for those locations where no high-resolution tiles are being decoded. Further details on late binding are provided in subclause 4.6.4.

An ISOBMFF Media Segment as specified in ISO/IEC 23009-1 includes one or more `MovieFragmentBox`es and one or more `MediaDataBox`es. It is allowed to split an ISOBMFF Media Segment into Subsegments, each containing one or more `MovieFragmentBox`es and one or more `MediaDataBox`es. A DASH client either requests entire Media Segments through their URLs indicated in the MPD or entire Subsegments through their byte ranges indicated in the `SegmentIndexBox`(es).

The duration of a Segment or a Subsegment can be too long for responding to a viewing orientation change with a new selection of OMAF tile tracks. To enable fine-grained requests, even down to a single picture interval, and to obtain the indexing data conveniently for all OMAF tile tracks, subclause 8.6 specifies the formats for the Initialization Segment for an OMAF base track, the Tile Index Segment, and the Tile Data Segment. The Initialization Segment for an OMAF base track contains not only a header for the OMAF base track but also a header for all the referenced OMAF tile tracks. The Tile Index Segment is an Index Segment as specified in ISO/IEC 23009-1 and contains `MovieFragmentBox`es for the OMAF base track and all the referenced OMAF tile tracks. A Tile Data Segment contains media data for the respective OMAF tile track.

4.6.2 Tile-based streaming with viewport-specific author-driven binding

When an SRQR descriptor is present in one or more Main Adaptation Sets, an OMAF player concludes that the content has been made available for viewport-specific author-driven binding. An OMAF player chooses a Main Adaptation Set that provides a higher quality for the viewport as indicated by the SRQR descriptor. When there are many Main Adaptation Sets suitable for the viewport, the following factors can be used for selecting between them:

— Codec and the coding profile and level. These pieces of information are carried in the @codecs MPD attribute. The decoding capacity of the client device can constrain which Main Adaptation Sets are decodable in real-time.

— The restricted video scheme type carried in the @codecs MPD attribute or the projection format carried in the PF descriptor, or both.

— OMAF video profile. The OMAF video profile for the Main Adaptation Set is conveyed in the @profiles MPD attribute.

— Relative picture quality in the viewport area as indicated by the SRQR descriptor, when sphRegionQuality@quality_ranking_local_flag is equal to 0.

— The projected picture resolution from which the content on the viewport originates, as indicated by the SRQR descriptor. An OMAF player can select such Main Adaptation Set, which provides a suitably high resolution in relation to the resolution of the viewport.

— The region covered by the high-quality content, as indicated by the SRQR descriptor, in relation to the viewport size and margins for potential viewing orientation changes.

When a Partial Adaptation Set contains multiple Representations, the following factors can be used for selecting between them:

— Relative picture quality among the Representations as indicated by the SRQR descriptors. Alternatively, the `@bandwidth` value or the `@qualityRanking` value can be used as an indication of the respective quality among the Representations in the same Partial Adaptation Set.

— The bitrates for the Representations, as indicated with the `@bandwidth` MPD attribute, in relation to the estimated throughput.

4.6.3 Tile-based streaming with free-viewport author-driven binding

When an SRQR descriptor is absent in one or more Main Adaptation Sets, an OMAF player concludes that the content has been made available for free-viewport author-driven binding. When there are many Main Adaptation Sets, the following factors can be used for selecting between them:

— Codec and the coding profile and level. These pieces of information are carried in the @codecs MPD attribute. The decoding capacity of the client device can constrain which Main Adaptation Sets are decodable in real-time.

— The restricted video scheme type carried in the @codecs MPD attribute or the projection format carried in the PF descriptor, or both.

— OMAF video profile. The OMAF video profile for the Main Adaptation Set is conveyed in the @profiles MPD attribute.

An OMAF player can select the OMAF tile tracks to be received as follows:

- If the Segment formats specified in subclause 8.6 are in use, an OMAF player can use the following factors to select a subset of the OMAF tile tracks to be received:
  - The `TrackBox`es of the OMAF tile tracks among which an OMAF player needs to select are present in the Initialization Segment of the Representation in the Main Adaptation Set.
  - The number of track references from the OMAF base track to OMAF tile tracks or to `'alte'` track groups indicates the number of OMAF tile tracks that an OMAF player needs to select.
  - The `ContentCoverageBox` or the `SphereRegionQualityRankingBox` can be used to conclude which sphere region is covered by the OMAF tile track.
  - An OMAF player aims at selecting OMAF tile tracks covering the viewport so that they have a higher quality than the other selected OMAF tile tracks. The `@bandwidth` value or the `@qualityRanking` value can be used as an indication of the respective quality among the Representations in the same Partial Adaptation Set and hence to select between collocated OMAF tile tracks. Alternatively or additionally, the quality ranking metadata in the instances of `SphereRegionQualityRankingBox`, when present, can be used.
- Otherwise, an OMAF player selects a Representation from each Partial Adaptation Set. An OMAF player aims at selecting OMAF tile tracks covering the viewport so that they have a higher quality than the other selected OMAF tile tracks. When a Partial Adaptation Set contains multiple Representations of OMAF tile tracks, the following factors can be used for selecting between them:
  - Relative picture quality among the Representations as indicated by the SRQR descriptors. Alternatively, the `@bandwidth` value or the `@qualityRanking` value can be used as an indication of the respective quality among the Representations in the same Partial Adaptation Set.
  - The bitrates for the Representations, as indicated with the `@bandwidth` MPD attribute, in relation to the estimated throughput.

4.6.4 Tile-based streaming with late binding

4.6.4.1 Strategy

The client determines which tiles to retrieve. It will typically retrieve the full omnidirectional video in a low quality and the high-resolution tiles to provide high quality in the viewport. Although the strategy used by an OMAF player to determine which tiles to select at which point in time is up to the implementation, the following is a non-exhaustive list of elements that may be used in this process:

- Current viewport orientation (yaw, pitch, roll)
- Current zoom level (if any)
- The decoding capacity of the client-device. On a device with limited decoding capability, the tile selection algorithm may decide to limit the number of tiles it will fetch from the highest quality level. Or, alternatively, decide to skip a certain quality layer completely, and use a only fetch tiles from a medium quality level.
- Random access points for each tile
- Available network bandwidth
- Measured network latency
- The number of quality levels, and the number of tiles in each of those quality levels, along with their respective tile sizes

 - Individual base tracks can be used to distinguish different quality levels.

A typical operation is to identify the tiles covering the current viewport (and the expected viewport in the next few hundred milliseconds). Based on the readings of the data generated by the user device, the client conceptually maps this target viewport onto the projected frame. The high-resolution tiles covering this area on the projected frame are then selected for retrieval. Upon reception, bitstream rewriting is performed on the retrieved tiles such that the rewritten bitstream contains the tiles for rendering the viewport, hence providing a partial view of the total projected frame. The client may, in addition, also decide to merge in the bitstream one or more low resolution tiles for covering the rest of the of projected frame.

4.6.4.2 Bitstream rewriting

4.6.4.2.1 General

A player can have many decoders at its disposal, a few, or only one. The most common case in devices at the time of writing this document is that a player has only one decoder, or just a few. To enable operation of devices that have only a single decoder, the bitstreams for the individual tiles need to be merged into a single bitstream.

Depending on device, viewport and network circumstances, it can be possible for a client deciding to select fewer tiles than are needed to satisfy the resolution the decoder has been initialized with, resulting in ‘empty’ tiles. In such a case, in order to still create a compliant bitstream, the client will have to fill those empty tiles with valid data. One method for doing so is to copy an already added tile as often as needed to make sure all tile ‘slots’ are filled. Another method is for a client to generate ‘dummy’ tile data.

4.6.4.2.2 HEVC bitstream rewriting

The bitstream rewriting process is not specified in this document, with the present subclause giving directions on how to achieve such a rewrite for HEVC. Implementers are referred to Rec. ITU-T H.265 | ISO/IEC 23008-2 for more details on how to write a conformant bitstream.

Generating a conformant video bitstream typically consists of two types of rewriting that need to be carried out:

a) Rewriting parameter sets in the Initialization Segment.

b) Rewriting the slice segment header.

Additionally, it may be required to remove start code emulation prevention bytes before rewriting the slice segment header and insert the start code emulation prevention bytes after rewriting the slice segment header.

4.7 Additional functionalities

In addition to omnidirectional video, images and audio, this document specifies the following additional functionalities:

- Multiple viewpoints. A viewpoint can be regarded as an omnidirectional camera. Viewpoints can be used e.g. for the following purposes:
  - Provide several user-switchable camera positions to view the content. For example, a sports event can be captured from different positions.
  - Express a storyline where the user is given the choice to select which storyline path is followed. Each temporally contiguous portion in the storyline is represented by a viewpoint.

The storage of viewpoints is specified in subclause 7.12.

— Overlays. Video, image, or timed text can be indicated to be overlays of the omnidirectional background visual video or image. Overlays can be indicated to have sphere-relative or viewport-relative positions. The storage of overlays is specified in subclause 7.14.

The use of the additional functionalities can be indicated using the toolset brands specified in Clause 12.

4.8 Conformance and interoperability

4.8.1 General

This document specifies several referenceable features that implementations and external specifications may use to achieve interoperability. These referenceable features include the following:

— Specification of a coordinate system for omnidirectional media (subclause 5.1), and equations for a conversion process between sets of coordinate axes of this coordinate system (subclause 5.3).

— Omnidirectional projection formats (subclause 5.2).

— Media profiles, as defined in subclause 4.8.2. Individual media profiles are specified in Clause 10.

— Presentation profiles, as defined in subclause 4.8.3. Individual presentation profiles are specified in Clause 11.

— Mapping of media and presentation profiles to DASH and CMAF are specified in Annex B and Annex C, respectively.

— Toolset brands, as defined in subclause 4.8.4. Individual toolset brands are specified in Clause 12.

— Region-wise packing formats (subclause 5.4).

— Scheme types for post-decoder processing of omnidirectional video (subclause 7.6.1).

— Specific types of sphere region timed metadata tracks (subclause 7.7).

However, the technologies and features specified in this document may be used independently. For example, boxes specified in this document may be used independently of other OMAF features.

NOTE For implementing a media profile, it is suggested to refer to the specification of that media profile in Clause 10 and read other parts of this document selectively by following the cross-references included in the specification of the media profile.

This document specifies code points that implementations and external specifications may use for referring to respective OMAF features. The code points are summarized in subclause 4.8.5.

4.8.2 Media profiles

4.8.2.1 General

A media profile for timed media is defined as requirements and constraints for a set of one or more ISOBMFF tracks of a single media type. The conformance of a set of one or more ISOBMFF tracks to a media profile is specified as a combination of:

— Specification of which sample entry type(s) are allowed, and which constraints and extensions are required in addition to those imposed by the sample entry type(s).

— Constraints on the samples of the tracks, typically expressed as constraints on the elementary stream contained within the samples of the tracks.

A media profile for static media is defined as requirements and constraints for a set of one or more ISOBMFF items of a single media type. The conformance of a set of one or more ISOBMFF items to a media profile is specified as a combination of:

- Specification of which item type(s) are allowed, and which constraints and extensions are required in addition to those imposed by the item type(s).
- Constraints on the content of the items, typically expressed as constraints on the elementary stream contained within the items.

The elementary stream constraints of a media profile may be indicated by a requirement to comply with a certain profile and level of the media coding specification, possibly including additional constraints and extensions, such as a requirement of the presence of certain information for rendering and presentation.

NOTE The use of the elementary stream constraints specified for a media profile can be referenced by implementations and external specifications independently of the ISOBMFF requirements and constraints specified for the media profile. However, the encapsulation and use of a media profile for other encapsulation formats is outside of the scope of this document.

4.8.2.2 File decoding process and file decoder requirements for video and image media profiles

Each video or image media profile specified in Clause 10 includes a file decoding process such that all file decoders that conform to the video or image media profile will produce numerically identical cropped decoded pictures when invoking the file decoding process associated with that video or image media profile for a set of ISOBMFF tracks conforming to the video media profile or a set of ISOBMFF image items conforming to the image media profile, respectively. A bitstream that conforms to the elementary stream constraints specified for the video or image media profile is reconstructed as an intermediate product of the file decoding process. The output of the file decoding process consists of all of the following:

- a list of decoded pictures;
- for projected omnidirectional video or image, all of the following rendering metadata:
  - the projection format of the associated projected pictures,
  - the frame packing format of the associated projected pictures (when applicable),
  - the region-wise packing information (when applicable),
  - the rotation information (when applicable);
- for fisheye omnidirectional video or image, fisheye-specific rendering metadata;
- for mesh omnidirectional video or image, all of the following metadata:
  - a 3D mesh, consisting of mesh elements,
  - a mapping of rectangular regions in the decoded pictures to mesh elements,
  - the rotation information (when applicable).

Video media profiles may specify constraints on when rendering metadata is allowed to change.

A file decoder conforms to the file decoding process requirements of this document when it complies with both of the following:

- The file decoder includes a conforming decoder that produces numerically identical cropped decoded pictures to those produced by the file decoding process specified for the video or image media profile in Clause 10 (with the correct output order or output timing, as specified in the video or image coding specification of the video or image media profile, respectively).

— The file decoder outputs rendering metadata that is equivalent to that produced by the file decoding process specified for the video or image media profile in Clause 10 (with the correct association of the rendering metadata to particular cropped decoded pictures, as specified in this document).

Figure 4 illustrates an example file decoder, its outputs as described above, and its relation to other parts of an OMAF player implementation.

A player claiming conformance to a video or image media profile shall include a file decoder complying with the file decoding process of that video or image media profile as specified above. Specifications of a media profile may include a subclause on expectations of a player operation, for example including recommendations for rendering.

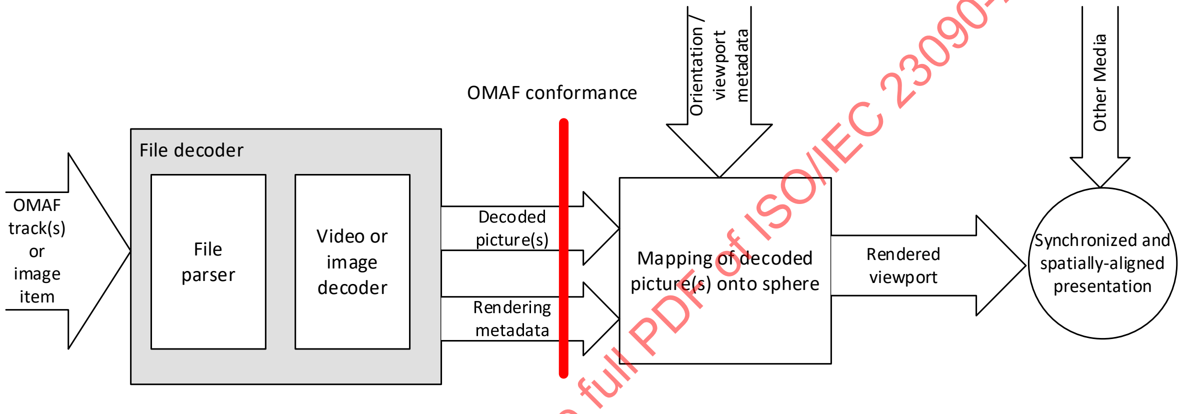

OMAF conformance
Orientation / viewport metadata
Other Media
File decoder
OMAF track(s) or image item
File parser
Video or image decoder
Decoded picture(s)
Rendering metadata
Mapping of decoded picture(s) onto sphere
Rendered viewport
Synchronized and spatially-aligned presentation

Figure 4 — Conformance points for OMAF video and image media profiles

4.8.2.3 File decoding process and file decoder requirements for audio media profiles

Each audio profile specified in Clause 10 includes a file decoding process such that all file decoders that conform to the audio profile will produce an output according to the specification of the file decoding process. For audio, the conformance of the output signal is defined by the file decoding process. The output of the file decoding process provides a signal that may be represented in the reference system.

4.8.3 Presentation profiles

A presentation profile is defined as requirements and constraints for an ISOBMFF file containing tracks or items of any number of media types. A specification of a presentation profile should refer to the specified media profiles and may include additional requirements or constraints. A file conforming to a presentation profile typically provides an omnidirectional audio-visual experience.

4.8.4 Toolset brands

A toolset brand facilitates a functionality that is considered additional compared to the basic operation of an OMAF player. A toolset brand should be specified so that the toolset brand can be used with any media coding format or media profile. A toolset brand is specified as requirements and constraints for a set of one or more ISOBMFF tracks and items. A file is indicated to conform to a toolset brand with a `FileTypeBox` containing the four-character code of the toolset brand among `compatible_brands`. The specification of a toolset brand may include requirements for an OMAF player complying with the toolset brand.

4.8.5 Summary of referenceable code points

4.8.5.1 Brands

ISO/IEC 14496-12 defines the concept of brands, which may be indicated in a file-specific manner in the `FileTypeBox` and in a track-specific manner in the `TrackTypeBox`. Brands are used in this document to indicate conformance to OMAF media profiles, OMAF presentation profiles, OMAF toolset brands, and CMAF Media Profiles that are defined in this document.

The brands specified in this document are listed in Table 2.

Table 2 — Brands specified in this document

Brand identifier	Subclause in this document	Name of the media profile or the presentation profile
`hevi`	10.1.2	HEVC-based viewport-independent OMAF video profile
`hevd`	10.1.3	HEVC-based viewport-dependent OMAF video profile
`avde`	10.1.4	AVC-based viewport-dependent OMAF video profile
`uhvi`	10.1.5	Unconstrained HEVC-based viewport-independent OMAF video profile
`adti`	10.1.5	Advanced tiling OMAF video profile
`siti`	10.1.6	Simple tiling OMAF video profile
`oabl`	10.2.2	OMAF 3D audio baseline profile
`oa2d`	10.2.3	OMAF 2D audio legacy profile
`heoi`	10.3.3	OMAF HEVC image profile
`jpoi`	10.3.4	OMAF legacy image profile
`ttml`	10.4.2	OMAF IMSC1 timed text profile
`ttwv`	10.4.3	OMAF WebVTT timed text profile
`ompp`	11.1	OMAF viewport-independent baseline presentation profile
`ovdp`	11.2	OMAF viewport-dependent baseline presentation profile
`ovly`	12.1	Overlay toolset brand
`vwpt`	12.2	Viewpoint toolset brand
`nlsl`	12.3	Non-linear storyline toolset brand
`cvid`	C.1.1	CMAF Media Profile for the HEVC-based viewport-independent OMAF video profile
`chev`	C.1.2	CMAF Media Profile for the HEVC-based viewport-dependent OMAF video profile

Brand identifier	Subclause in this document	Name of the media profile or the presentation profile
`cuvd`	C.1.3	CMAF Media Profile for the unconstrained HEVC-based viewport-independent OMAF video profile
`cabl`	C.2.1	CMAF Media Profile for OMAF 3D audio baseline profile

4.8.5.2 Uniform resource names

The URNs specified in this document are listed in Table 3.

Table 3 — URNs specified in this document

URN	Subclause in this document	Description
`urn:mpeg:mpegI:omaf:2017`	8.2.2	Namespace for the XML elements and attributes specified in Edition 1 of this document
`urn:mpeg:mpegI:omaf:2017:pf`	8.3.2	Scheme identifier for the omnidirectional projection type DASH MPD descriptor
`urn:mpeg:mpegI:omaf:2017:rwpk`	8.3.3	Scheme identifier for the region-wise packing type DASH MPD descriptor
`urn:mpeg:mpegI:omaf:2017:cc`	8.3.4	Scheme identifier for the content coverage DASH MPD descriptor
`urn:mpeg:mpegI:omaf:2017:srqr`	8.3.5	Scheme identifier for the spherical region-wise quality ranking DASH MPD descriptor
`urn:mpeg:mpegI:omaf:2017:2dqr`	8.3.6	Scheme identifier for the 2D region-wise quality ranking DASH MPD descriptor
`urn:mpeg:mpegI:omaf:2017:fomv`	8.3.7	Scheme identifier for the fisheye omnidirectional video DASH MPD descriptor
`urn:mpeg:mpegI:omaf:2020`	8.5.1	Namespace for the XML elements and attributes specified in Edition 2 of this document
`urn:mpeg:mpegI:omaf:2020:assoc`	8.5.2	Scheme identifier for the association descriptor
`urn:mpeg:mpegI:omaf:2020:vwpt`	8.5.3	Scheme identifier for the viewpoint information descriptor
`urn:mpeg:mpegI:omaf:2020:ovly`	8.5.5	Scheme identifier for the overlay information descriptor
`urn:mpeg:mpegI:omaf:2020:etgb`	8.5.6	Scheme identifier for the entity to group descriptor

URN	Subclause in this document	Description
`urn:mpeg:mpegI:omaf:dash:profile:indexed-isobmff:2020`	B.1.4.1	DASH profile identifier indicating the use of the segment formats specified in subclause 8.5

4.8.5.3 Restricted scheme types

The restricted scheme types specified in this document are listed in Table 4.

Table 4 — Restricted scheme types specified in this document

Restricted scheme type	Subclause in this document	Description
`podv`	7.6.1.2	Projected omnidirectional video; an open-ended restricted scheme type for omnidirectional video based on a projection
`erpv`	7.6.1.3	Equirectangular projected video; a closed restricted scheme type for omnidirectional video based on the equirectangular projection and essentially no region-wise packing.
`ercm`	7.6.1.4	Packed equirectangular or cubemap projected video; a closed restricted scheme type for omnidirectional video based on either the equirectangular projection or the cubemap projection and any region-wise packing.
`fodv`	7.6.1.5	Fisheye omnidirectional video; an open-ended restricted scheme type for omnidirectional video based on video captured by one or more fisheye camera lenses and not based on a projection
`ecov`	7.6.1.6	Like `'ercm'` but overlay information may additionally be present.
`erc2`	7.6.1.7	Like `'ercm'` but the region-wise packing format of the track containing the `'erc2'` scheme type is derived from the collection of the instances of `RegionWisePackingBox` of the tracks referenced by the track containing the `'erc2'` scheme type.
`modv`	7.6.1.8	Mesh omnidirectional video; an open-ended restricted scheme type for omnidirectional video based on a 3D mesh
`meov`	7.6.1.9	Closed scheme type for mesh omnidirectional video

4.8.5.4 Sample entry types

The sample entry types specified in this document are listed in Table 5.

Table 5 — Sample entry types specified in this document

Sample entry type	Subclause in this document	Description
`invo`	7.7.4	For use with a timed metadata track containing the initial viewing orientation timed metadata
`rcvp`	7.7.5	For use with a timed metadata track containing the recommended viewport timed metadata without indicating a viewpoint
`rvp2`	7.7.5	For use with a timed metadata track containing the recommended viewport timed metadata concerning one or more indicated viewpoints
`ttsl`	7.7.7	For use with a timed metadata track containing the timed text sphere location timed metadata
`stmd`	7.11	For use with a timed metadata track containing ERP region metadata
`dyvp`	7.12.3.1	For use with a timed metadata track indicating the viewpoint parameters that are dynamically changing over time
`invp`	7.12.3.2	For use with a timed metadata track indicating the initial viewpoint that should be used
`ocpc`	7.12.3.3	For use with a timed metadata track indicating information on object centre points correspondence between viewpoints
`dyol`	7.14.6	For use with a timed metadata track indicating overlays that are active only at particular times and dynamically changing overlay parameters
`vrsp`	7.15.5	For use with a timed metadata track indicating viewing space

4.8.5.5 Box types

The box types specified in this document are listed in Table 6.

Table 6 — Box types specified in this document and their relation to boxes not specified in this document

Box types, structure and cross-reference.												
moov											ISOBMFF	*container for all the metadata*
	trak										ISOBMFF	*container for an individual track or stream*
		tref									ISOBMFF	*track reference container*
			cdtg								7.1.2.1	*track reference for describing the referenced tracks and track groups collectively*
			shsc								7.1.5	*track reference to a shadow sync sample track*
		trgr									ISOBMFF	*track grouping indication*
			2dsr								7.1.4	*spatial relationship 2D description box*
				2dss							7.1.4	*spatial relationship 2D source box*
				sprg							7.1.4	*sub-picture region box*
				coif							7.13.3	*composition information box*
		ttyp									ISOBMFF	*track type box*
		mdia									ISOBMFF	*container for the media information in a track*
			minf								ISOBMFF	*media information container*
				stbl							ISOBMFF	*sample table box, container for the time/space map*
					stsd						ISOBMFF	*sample descriptions (codec types, initialization etc.)*
						-					ISOBMFF	*sample entry or restricted sample entry*
							rinf				ISOBMFF	*restricted scheme info box*
								schi			ISOBMFF	*scheme information box*
									povd		7.6.2	*projected omnidirectional video box*
										prfr	7.6.2	*projection format box*
										rwpk	7.6.4	*region-wise packing box*
										rotn	7.6.5	*rotation box*
										covi	7.6.6	*coverage information box*
										ovly	7.14.4	*overlay configuration box*
									fovd		7.6.3	*fisheye omnidirectional video box*
										fovi	7.6.3	*fisheye video essential info box*
										fvsi	7.6.3	*fisheye video supplemental info box*
									movd		7.6.7	*mesh omnidirectional video box*
										mesh	7.6.8	*mesh box*
										rotn	7.6.5	*rotation box*
						-					7.7.2	*sphere region sample entry*
							rosc				7.7.2	*sphere region configuration box*
						rcvp					7.7.5	*recommended viewport sample entry*
							rvif				7.7.5	*recommended viewport information box*
						-					ISOBMFF	*visual sample entry*

Box types, structure and cross-reference.												
							srqr				7.8.2	*sphere region quality ranking box*
							2dqr				7.8.3	*2D region quality ranking box*
							ovly				7.14.4	*overlay configuration box*
							vssn				7.15.3	*viewing space box*
						stpp					ISOBMFF	*XML subtitle sample entry*
							otcf				7.10.2	*OMAF timed text configuration box*
						wvtt					ISO/IEC 14496-30	*WebVTT sample entry*
							otcf				7.10.2	*OMAF timed text configuration box*
meta											ISOBMFF	*Metadata*
	iprp										ISOBMFF	*item properties box*
		ipco									ISOBMFF	*item property container box*
			prfr								7.9.3	*projection format item property*
			fovi								7.9.4	*essential fisheye image item property*
			fvsi								7.9.5	*supplemental fisheye image item property*
			rwpk								7.9.6	*region-wise packing item property*
			rotn								7.9.7	*rotation item property*
			covi								7.9.8	*coverage information item property*
			stvi								7.9.2	*frame packing item property* NOTE `FramePackingProperty` has the same box type and syntax as `StereoVideoBox` specified in ISO/IEC 14496-12.
			iivo								7.9.9	*initial viewing orientation item property*
			ovly								7.14.5	*overlay item property*
			vssn								7.15.4	*viewing space item property*
meof											7.1.6	*media offset box*

4.8.5.6 Track grouping types

The track grouping types specified in this document are listed in Table 7.

Table 7 — Track grouping types specified in this document

Track grouping type	Subclause in this document	Track grouping name
`2dsr`	7.1.4.2	Two dimensional spatial relationship track grouping

4.8.5.7 Entity grouping types

The entity grouping types specified in this document are listed in Table 8.

Table 8 — Entity grouping types specified in this document

Entity grouping type	Subclause in this document	Entity grouping name
vipo	7.12.2	Viewpoint entity grouping
oval	7.14.7.1	Grouping of overlays that are alternatives for switching
ovbg	7.14.7.2	Grouping of overlays and background visual media that are intended to be presented together

4.8.5.8 Sample grouping types

The sample grouping types specified in this document are listed in Table 9.

Table 9 — Sample grouping types specified in this document

Sample grouping type	Subclause in this document	Sample grouping name
2dsr	7.1.4.3	Two dimensional spatial relationship sample grouping
tmsh	7.16.2	Tile mesh sample grouping

5 Omnidirectional video projection and region-wise packing

5.1 Coordinate system

The coordinate system consists of a unit sphere and three coordinate axes, namely the X (back-to-front) axis, the Y (lateral, side-to-side) axis, and the Z (vertical, up) axis, where the three axes cross at the centre of the sphere.

The location of a point on the sphere is identified by a pair of sphere coordinates azimuth (ϕ) and elevation (θ).

Figure 5 specifies the relation of the sphere coordinates azimuth (ϕ) and elevation (θ) to the X, Y, and Z coordinate axes.

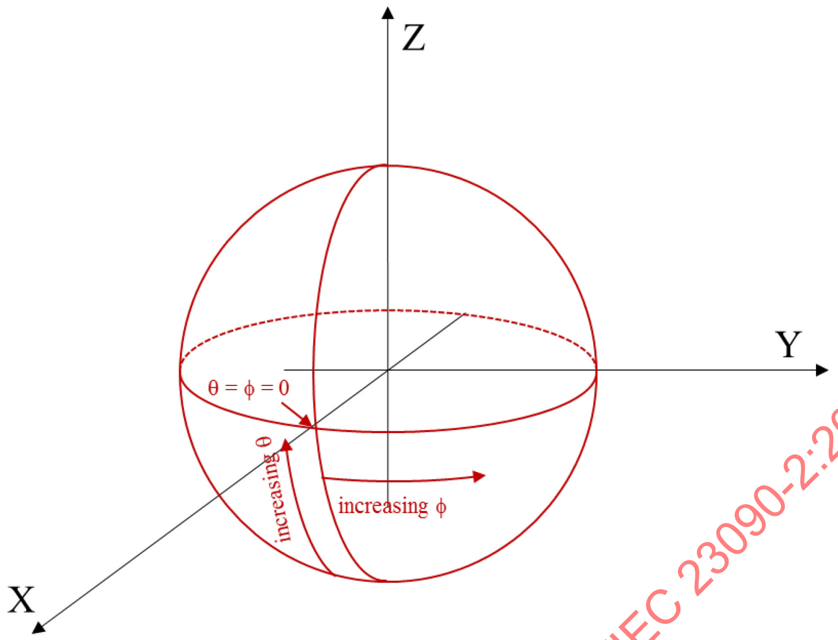


Figure 5 — Coordinate axes and their relation to the sphere coordinates

NOTE The specified coordinate system is the same as in ISO/IEC 23008-3.

The value ranges of azimuth is −180.0, inclusive, to 180.0, exclusive, degrees. The value range of elevation is −90.0 to 90.0, inclusive, degrees.

5.2 Omnidirectional projection formats

5.2.1 General

Subclause 5.2 specifies the inverse of the projection process for remapping of one sample location of a monoscopic projected luma picture onto a position on the unit sphere identified by a pair of azimuth and elevation coordinates.

The omnidirectional projection formats specified in this document are identified by a 5-bit unsigned integer value. Table 10 specifies the omnidirectional projection format identifier values and provides references to the subclauses in which the respective inverse projection processes are specified.

Table 10 — Omnidirectional projection formats

Identifier value	Omnidirectional projection	Subclause
0	Equirectangular projection	5.2.2
1	Cubemap projection	5.2.3
2..31	Reserved	N/A

5.2.2 Equirectangular projection for one sample location

Inputs to this process are:

— pictureWidth and pictureHeight, which are the width and height, respectively, of a monoscopic projected luma picture, in relative projected picture sample units, and

— the centre point of a sample location (hPos, vPos) along the horizontal and vertical axes, respectively, where hPos and vPos are in relative projected picture sample units and may have non-integer real values.

Outputs of this process are:

— sphere coordinates (ϕ, θ) for the sample location in degrees relative to the coordinate axes specified in subclause 5.1.

The sphere coordinates (ϕ, θ) for the luma sample location, in degrees, are given by the following equations:

$$\phi = (0.5 - \text{hPos} \div \text{pictureWidth}) * 360 \qquad (5)$$
$$\theta = (0.5 - \text{vPos} \div \text{pictureHeight}) * 180$$

Figure 6 illustrates the azimuth and elevation ranges of a monoscopic projected picture with the equirectangular projection.

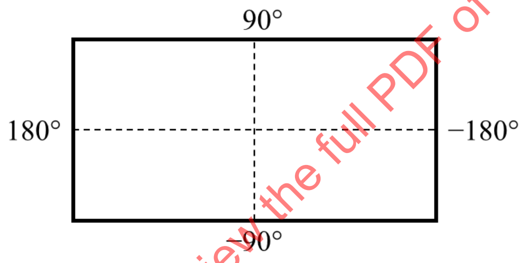


Figure 6 — Azimuth and elevation ranges of the monoscopic projected picture of the equirectangular projection

NOTE 1 Since an input to this process is the centre point of a sample location and the width and height of the sample are non-zero, the output ϕ is never equal to −180° or 180° and the output θ is never equal to −90° or 90°.

NOTE 2 The monoscopic projected picture represents the inside surface of the unit sphere observed from the origin of the coordinate system. Thus, the azimuth decreases from left to right.

5.2.3 Cubemap projection for one sample location

Inputs to this process are:

— pictureWidth and pictureHeight, which are the width and height, respectively, of a monoscopic projected luma picture, in relative projected picture sample units, and

— the centre point of a sample location (hPos, vPos) along the horizontal and vertical axes, respectively, where hPos and vPos are in relative projected picture sample units and may have non-integer real values.

Outputs of this process are:

— sphere coordinates (ϕ, θ) for the sample location in degrees relative to the coordinate axes specified in subclause 5.1.

Figure 7 illustrates the cube face arrangement in the projected picture of the cubemap projection format and the mapping of the cube faces onto the coordinate axes specified in subclause 5.1, where PX, NX, PY, NY, PZ, and NZ denote positive X, negative X, positive Y, negative Y, positive Z, and negative Z, respectively.

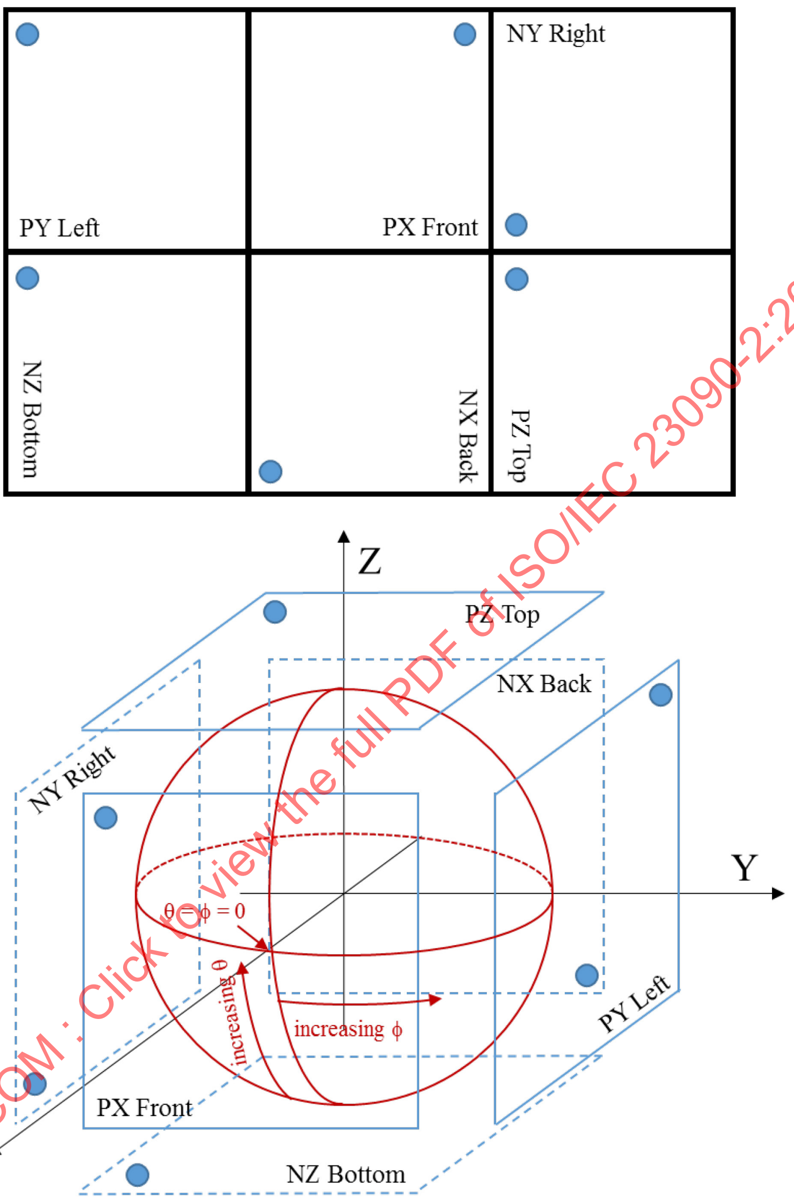


Figure 7 — Relation of the cube face arrangement of the projected picture to the sphere coordinates

The values of pictureWidth and pictureHeight shall be such that pictureWidth is a multiple of 3, pictureHeight is a multiple of 2, and pictureWidth / 3 is equal to pictureHeight / 2.

The sphere coordinates (ϕ, θ) for the luma sample location, in degrees, are given by the following equations:

```
lw = pictureWidth / 3
lh = pictureHeight / 2
tmpHorVal = hPos − Floor( hPos ÷ lw ) * lw
tmpVerVal = vPos − Floor( vPos ÷ lh ) * lh
hPos′ = −( 2 * tmpHorVal ÷ lw ) + 1
vPos′ = −( 2 * tmpVerVal  ÷ lh ) + 1
w = Floor( hPos ÷ lw )
```

```
h = Floor( vPos ÷ lh )
if( w  = =  1  &&  h  = =  0 ) { // positive x front face
      x = 1.0
      y = hPos′
      z = vPos′
} else if( w  = =  1  &&  h  = =  1 ) { // negative x back face
      x = −1.0
      y = −vPos′
      z = −hPos′
} else if( w  = =  2  &&  h  = =  1 ) { // positive z top face                    (6)
      x = −hPos′
      y = −vPos′
      z = 1.0
} else if( w  = =  0  &&  h  = =  1 ) { // negative z bottom face
      x = hPos′
      y = −vPos′
      z = −1.0
} else if( w  = =  0  &&  h  = =  0 ) { // positive y left face
      x = −hPos′
      y = 1.0
      z = vPos′
} else { // ( w  = =  2  &&  h  = =  0 ), negative y right face
      x = hPos′
      y = −1.0
      z = vPos′
}
$\phi$ = Atan2( y, x ) * 180 ÷ π
$\theta = \text{Asin}\left(z \div \sqrt{x^2 + y^2 + z^2}\right) * 180 \div \pi$
```

5.3 Conversion from the local coordinate axes to the global coordinate axes

Inputs to this process are:

— `rotation_yaw` (α_d), `rotation_pitch` (β_d), `rotation_roll` (γ_d), all in units of degrees, where `rotation_yaw` (α_d) and `rotation_roll` (γ_d), are in the range of −180.0, inclusive, to 180.0, exclusive, and `rotation_pitch` (β_d) is in the range of −90.0 to 90.0, inclusive, and

— sphere coordinates (ϕ_d, θ_d) relative to the local coordinate axes.

Outputs of this process are:

— sphere coordinates (ϕ', θ') in degrees relative to the global coordinate axes.

This process specifies rotations around the three axes of the coordinate system of subclause 5.1, where yaw (α_d) expresses a rotation around the Z axis, pitch (β_d) rotates around the Y axis, and roll (γ_d) rotates around the X axis. Rotations are extrinsic, i.e. around X, Y, and Z fixed reference axes. The angles increase clockwise when looking from the origin towards the positive end of an axis, as illustrated in Figure 8.

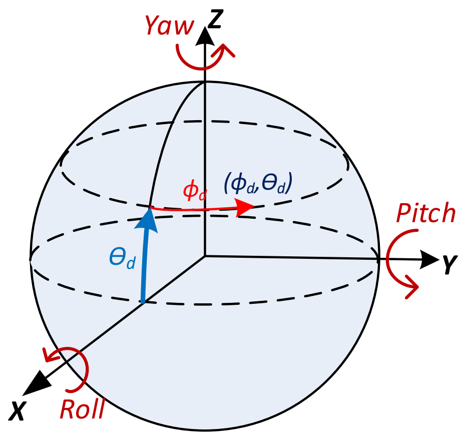


Figure 8 — Illustration of the directions of the yaw, pitch, and roll rotations

When any of the yaw (α_d), pitch (β_d) and roll (γ_d) rotation angles is not equal to zero, an OMAF player needs to apply the sphere rotation process specified in this subclause to convert the local coordinate axes to the global coordinate axes.

It is assumed that the global coordinate systems for different media types were made aligned during content production.

The outputs are derived as follows:

$$\begin{aligned}
&\phi = \phi_d * \pi \div 180\\
&\theta = \theta_d * \pi \div 180\\
&\alpha = \alpha_d * \pi \div 180\\
&\beta = \beta_d * \pi \div 180\\
&\gamma = \gamma_d * \pi \div 180\\
&x_1 = \mathrm{Cos}(\ \phi\ ) * \mathrm{Cos}(\ \theta\ )\\
&y_1 = \mathrm{Sin}(\ \phi\ ) * \mathrm{Cos}(\ \theta\ )\\
&z_1 = \mathrm{Sin}(\ \theta\ )\\
&x_2 = \mathrm{Cos}(\ \beta\ ) * \mathrm{Cos}\ (\ \alpha\ ) * x_1 - \mathrm{Cos}(\ \beta\ ) * \mathrm{Sin}(\ \alpha\ ) * y_1 + \mathrm{Sin}(\ \beta\ ) * z_1 \qquad (7)\\
&y_2 = (\ \mathrm{Cos}(\ \gamma\ ) * \mathrm{Sin}(\ \alpha\ ) + \mathrm{Sin}(\ \gamma\ ) * \mathrm{Sin}(\ \beta\ ) * \mathrm{Cos}(\ \alpha\ )\ ) * x_1 +\\
&\qquad (\ \mathrm{Cos}(\ \gamma\ ) * \mathrm{Cos}(\ \alpha\ ) - \mathrm{Sin}(\ \gamma\ ) * \mathrm{Sin}(\ \beta\ ) * \mathrm{Sin}(\ \alpha\ )\ ) * y_1 -\\
&\qquad \mathrm{Sin}(\ \gamma\ ) * \mathrm{Cos}(\ \beta\ ) * z_1\\
&z_2 = (\ \mathrm{Sin}(\ \gamma\ ) * \mathrm{Sin}(\ \alpha\ ) - \mathrm{Cos}(\ \gamma\ ) * \mathrm{Sin}(\ \beta\ ) * \mathrm{Cos}(\ \alpha\ )\ ) * x_1 +\\
&\qquad (\ \mathrm{Sin}(\ \gamma\ ) * \mathrm{Cos}(\ \alpha\ ) + \mathrm{Cos}(\ \gamma\ ) * \mathrm{Sin}(\ \beta\ ) * \mathrm{Sin}(\ \alpha\ )\ ) * y_1 +\\
&\qquad \mathrm{Cos}(\ \gamma\ ) * \mathrm{Cos}(\ \beta\ ) * z_1\\
&\phi' = \mathrm{Atan2}(\ y_2, x_2\ ) * 180 \div \pi\\
&\theta' = \mathrm{Asin}(\ z_2\ ) * 180 \div \pi
\end{aligned}$$

5.4 Region-wise packing formats

5.4.1 General

Subclause 5.4 specifies the inverse processes of the region-wise packing for remapping of a luma sample location in a packed region onto a luma sample location of the corresponding projected region.

The inverse region-wise packing processes specified in this document are identified by a 4-bit unsigned integer value. Table 11 specifies the region-wise packing format identifier values and provides references to the subclauses in which the respective inverse processes are specified.

Table 11 — Region-wise packing formats

Identifier value	Region-wise packing format	Subclause
0	Rectangular region-wise packing	5.4.2
1..15	Reserved	N/A

5.4.2 Conversion of one sample location for rectangular region-wise packing

This subclause specifies the inverse of the rectangular region-wise packing process for remapping of a luma sample location in a packed region onto a luma sample location of the corresponding projected region.

Inputs to this process are:

— sample location (x, y) within the packed region, where x and y are in relative packed picture sample units, while the sample location is at an integer sample location within the packed picture,

— the width and the height (projRegWidth, projRegHeight) of the projected region, in relative projected picture sample units,

— the width and the height (packedRegWidth, packedRegHeight) of the packed region, in relative packed picture sample units,

— transform type (transformType), and

— offset values for the sampling position (offsetX, offsetY) in the range of 0, inclusive, to 1, exclusive, in horizontal and vertical relative packed picture sample units, respectively.

NOTE offsetX and offsetY both equal to 0.5 indicate a sampling position that is in the centre point of a sample in packed picture sample units.

Outputs of this process are:

— the centre point of the sample location (hPos, vPos) within the projected region, where hPos and vPos are in relative projected picture sample units and may have non-integer real values.

The outputs are derived as follows:

```
if( transformType = = 0 || transformType = = 1 || transformType = = 2 || transformType = = 3 ) {
    horRatio = projRegWidth ÷ packedRegWidth
    verRatio = projRegHeight ÷ packedRegHeight
} else if ( transformType = = 4 || transformType = = 5 || transformType = = 6 ||
    transformType = = 7 ) {
    horRatio = projRegWidth ÷ packedRegHeight
    verRatio = projRegHeight ÷ packedRegWidth
}
if( transformType = = 0 ) {
    hPos = horRatio * ( x + offsetX )
    vPos = verRatio * ( y + offsetY )
} else if ( transformType = = 1 ) {
    hPos = horRatio * ( packedRegWidth − x − offsetX )                    (8)
    vPos = verRatio * ( y + offsetY )
} else if ( transformType = = 2 ) {
    hPos = horRatio * ( packedRegWidth − x − offsetX )
    vPos = verRatio * ( packedRegHeight − y − offsetY )
} else if ( transformType = = 3 ) {
        hPos = horRatio * ( x + offsetX )
        vPos = verRatio * ( packedRegHeight − y − offsetY )
    } else if ( transformType  = =  4 ) {
        hPos = horRatio * ( y + offsetY )
        vPos = verRatio * ( x + offsetX )
    } else if ( transformType  = =  5 ) {
        hPos = horRatio * ( y + offsetY )
        vPos = verRatio * ( packedRegWidth − x − offsetX )
    } else if ( transformType  = = 6 ) {
        hPos = horRatio * ( packedRegHeight − y − offsetY )
        vPos = verRatio * ( packedRegWidth − x − offsetX )
    } else if ( transformType  = =  7 ) {
        hPos = horRatio * ( packedRegHeight − y − offsetY )
        vPos = verRatio * ( x+ offsetX )
    }
```

6 Fisheye omnidirectional video

6.1 General

Without the *projection* and *region-wise packing* processes specified in Clause 5, multiple circular images captured by fisheye cameras may be directly projected onto a picture, which consists of fisheye omnidirectional video. In an OMAF player, the decoded fisheye omnidirectional video may be stitched and rendered according to the user's intended viewport using the signalled fisheye video parameters, including:

— Region information of circular images in the coded picture,

— Field of view and camera parameters of fisheye lens,

— Lens distortion correction (LDC) parameters with local variation of FOV, and

— Lens shading compensation (LSC) parameters with RGB gains.

Essential fisheye video parameters for enabling stitching and rendering at the OMAF player are specified in `FisheyeVideoEssentialInfoStruct()` in subclause 6.2. For high quality rendering and efficient delivery of fisheye video, supplemental fisheye video parameters are specified in `FisheyeVideoSupplementalInfoStruct()` in subclause 6.3.

6.2 The `FisheyeVideoEssentialInfoStruct()` syntax structure

6.2.1 Syntax

```
aligned(8) class FisheyeVideoEssentialInfoStruct() {
    unsigned int(3) view_dimension_idc;
    bit(21) reserved = 0;
    unsigned int(8) num_circular_images;
    for (i=0; i< num_circular_images; i++) {
       unsigned int(32) circular_image_centre_x;
       unsigned int(32) circular_image_centre_y;
       unsigned int(32) rect_region_top;
       unsigned int(32) rect_region_left;
       unsigned int(32) rect_region_width;
       unsigned int(32) rect_region_height;
```

```
        unsigned int(32) circular_image_radius;
        unsigned int(32) scene_radius;
        signed int(32) camera_centre_azimuth;
        signed int(32) camera_centre_elevation;
        signed int(32) camera_centre_tilt;
        unsigned int(32) camera_centre_offset_x;
        unsigned int(32) camera_centre_offset_y;
        unsigned int(32) camera_centre_offset_z;
        unsigned int(32) field_of_view;
        bit(16) reserved = 0;
        unsigned int(16) num_polynomial_coeffs;
        for (j=0; j< num_polynomial_coeffs; j++)
            signed int(32) polynomial_coeff;
    }
}
```

6.2.2 Semantics

view_dimension_idc indicates that the syntax element values of this FisheyeVideoEssentialInfoStruct() syntax structure have been constrained as specified in Table 12. The indicated constraints shall be obeyed in the syntax element values of this FisheyeVideoEssentialInfoStruct() syntax structure.

Table 12 — Constraints implied by view_dimension_idc

view_dimension_idc	Constraints
0	num_circular_images is equal to 2. The values of camera_centre_azimuth, camera_centre_elevation, camera_centre_tilt, camera_centre_offset_x, camera_centre_offset_y, and camera_centre_offset_z are such that the circular images have aligned optical axes and face opposite directions. The sum of field_of_view values is greater than or equal to 360 * 2^{16}.
1	num_circular_images is equal to 2. The values of camera_centre_azimuth, camera_centre_elevation, camera_centre_tilt, camera_centre_offset_x, camera_centre_offset_y, and camera_centre_offset_z are such that the circular images have parallel optical axes that are orthogonal to the line intersecting the camera centre points. The camera corresponding to i equal to 0 is the left view.

view_dimension_idc	Constraints
2	num_circular_images is equal to 2. The values of camera_centre_azimuth, camera_centre_elevation, camera_centre_tilt, camera_centre_offset_x, camera_centre_offset_y, and camera_centre_offset_z are such that the circular images have parallel optical axes that are orthogonal to the line intersecting the camera centre points. The camera corresponding to i equal to 0 is the right view.
3 to 6, inclusive	Reserved.
7	No additional constraints are implied for the syntax element values within this FisheyeVideoEssentialInfoStruct() syntax structure.

num_circular_images specifies the number of circular images in the cropped output picture of each coded picture this structure applies to. Typically, the value is equal to 2, but other non-zero values are also possible.

circular_image_centre_x is a fixed-point 16.16 value that specifies the horizontal coordinate, in luma samples, of the centre of the circular image in each coded picture this structure applies to, relative to the top-left corner of the coded picture.

circular_image_centre_y is a fixed-point 16.16 value that specifies the vertical coordinate, in luma samples, of the centre of the circular image in each coded picture this structure applies to, relative to the top-left corner of the coded picture.

rect_region_top, rect_region_left, rect_region_width, and rect_region_height specify the coordinates of the top-left corner and the width and height of the rectangular region that contains the circular image, which may or may not be cropped. These values are specified in units of luma samples.

circular_image_radius is a fixed-point 16.16 value that specifies the radius, in luma samples, of the circular image that is defined as a length from the centre of the circular image specified by circular_image_centre_x and circular_image_centre_y to the outermost pixel boundary of the circular image, that corresponds to the maximum field of view of the fisheye lens, specified by field_of_view.

The active area of a circular image is defined by the intersection of the rectangular region, specified by rect_region_top, rect_region_left, rect_region_width, and rect_region_height, and the circular image specified by circular_image_centre_x, circular_image_centre_y, and circular_image_radius.

The active area of each circular image shall contain at least one sample location. There shall not be any sample location that is within more than one active area.

scene_radius is a fixed-point 16.16 value that specifies the radius, in luma samples, of a circular region within the active area of the circular image, where the obstruction, such as the camera body, is not included in the region specified by circular_region_centre_x, circular_region_centre_y, and scene_radius. The enclosed area is the suggested area for stitching as recommended by the content provider.

camera_centre_azimuth and camera_centre_elevation indicate the spherical coordinates that correspond to the centre of the circular image. camera_centre_azimuth shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive. camera_centre_elevation shall be in the range of $-90 * 2^{16}$ to $90 * 2^{16}$, inclusive.

camera_centre_tilt indicates the tilt angle of the sphere region that corresponds to the circular image. camera_centre_tilt shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive.

camera_centre_offset_x, camera_centre_offset_y and camera_centre_offset_z are fixed-point 16.16 values that indicate the XYZ offset values, in millimetres, of the focal centre of the fisheye camera lens corresponding to the circular image, from the focal centre origin of the overall fisheye camera configuration as illustrated in Figure 9.

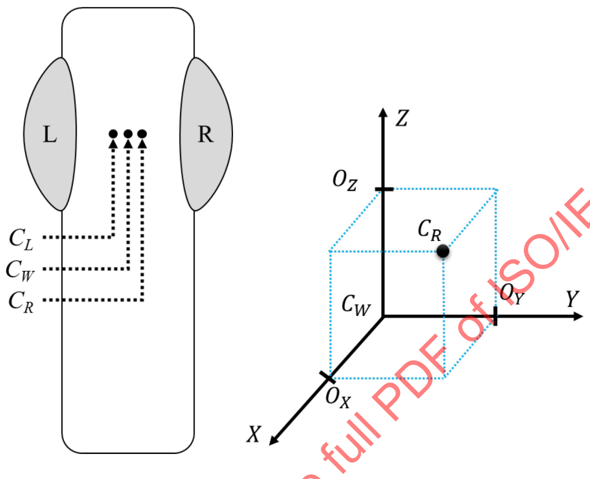


Figure 9 – Illustration of camera_centre_offset_x (O_X), camera_centre_offset_y (O_Y) and camera_centre_offset_z (O_Z) of the focal centre (C_L or C_R) of the fisheye camera lens (L or R, respectively) relative to the focal centre origin of the overall fisheye camera configuration (C_W)

field_of_view is a fixed-point 16.16 value that specifies the field of view of the fisheye lens corresponding to the circular image, in degrees. A typical value for a hemispherical fisheye lens is 180.0 degrees.

num_polynomial_coeffs and polynomial_coeff are lens distortion parameters that define the curve function that maps the normalized distance (r) of a luma sample (x_p) from the centre of the circular image to the angle (θ) of a sphere coordinate from the normal vector of a nominal imaging plane that passes through the centre of the sphere coordinate system for the active area of the circular image, where the angle is in units of radians, as shown in Figure 10.

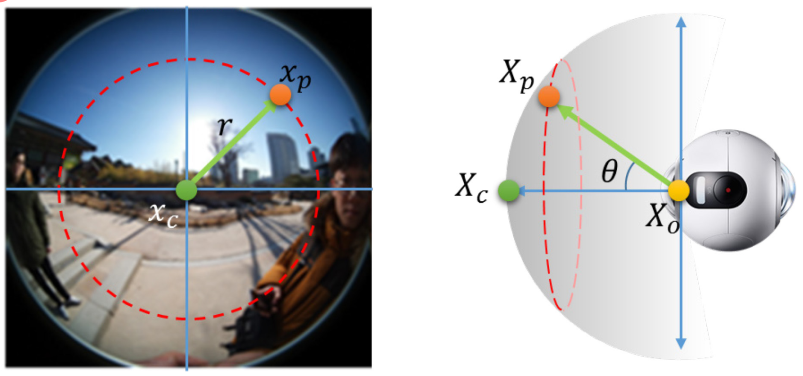


Figure 10 — Illustration of the relation between the normalized distance (r) and the angular value (θ) of distortion function

`num_polynomial_coeffs` is an integer that specifies the number of polynomial coefficients for the circular image. This is the maximum order of the polynomial plus 1. The value of `num_polynomial_coeffs` shall be in the range of 0 to 8, inclusive. Values of `num_polynomial_coeffs` greater than 8 are reserved.

The instances of polynomial `polynomial_coeff` are fixed-point 8.24 polynomial coefficient values that describes the curve function using the following polynomial equation:

$$\theta = \sum_{j=0}^{N_i - 1} p_j * r^j \tag{9}$$

where p_j and N_i are represented by `polynomial_coeff` and `num_polynomial_coeffs`, respectively.

6.3 The `FisheyeVideoSupplementalInfoStruct()` syntax structure

6.3.1 Syntax

```
aligned(8) class FisheyeVideoSupplementalInfoStruct(container_box_version) {
    unsigned int(8) num_circular_images;
    bit(19) reserved = 0;
    unsigned int(1) entrance_pupil_flag;
    unsigned int(1) flip_info_flag;
    unsigned int(1) camera_intrinsic_flag;
    unsigned int(1) local_fov_flag;
    unsigned int(1) deadzone_flag;

    for (i=0; i<num_circular_images; i++) {
       if (container_box_version > 0) {
           unsigned int(4) lens_projection_type;
           bit(4) reserved = 0;
       }
       if (entrance_pupil_flag == 1 ) {
           unsigned int(16) num_ep_coeffs;
           for (j=0; j<num_ep_coeffs; j++)
               signed int(32) ep_coeff;
       }
       if (flip_info_flag == 1) {
           bit(30) reserved = 0;
           unsigned int(2) image_flip;
       }
       if (camera_intrinsic_flag == 1) {
           unsigned int(32) image_scale_axis_angle;
           unsigned int(32) image_scale_x;
           unsigned int(32) image_scale_y;
           bit(16) reserved = 0;
           unsigned int(16) num_polynomial_coefs_lsc;
           for (j=0; j<num_polynomial_coefs_lsc; j++) {
               signed int (32) polynomial_coef_k_lsc_r;
               signed int (32) polynomial_coef_k_lsc_g;
               signed int (32) polynomial_coef_k_lsc_b;
           }
       }
       if (local_fov_flag == 1) {
           unsigned int (16) num_angle_for_displaying_fov;
           bit(16) reserved = 0;
           for (j=0; j<num_angle_for_displaying_fov; j++) {
               unsigned int(32) displayed_fov;
               unsigned int(32) overlapped_fov;
           }
```

```
            bit(16) reserved = 0;
            unsigned int (16) num_local_fov_region;
            for (j=0; j<num_local_fov_region; j++) {
                unsigned int(32) start_radius;
                unsigned int(32) end_radius;
                signed int(32) start_angle;
                signed int(32) end_angle;
                unsigned int(32) radius_delta;
                signed int(32) angle_delta;
                for (rad=start_radius; rad<=end_radius; rad+=radius_delta)
                    for (ang=start_angle; ang<= end_angle; ang+=angle_delta)
                        unsigned int(32) local_fov_weight;
            }
        }
    }
    if (deadzone_flag == 1) {
        bit(24) reserved = 0;
        unsigned int(8) num_deadzones;
        for (j=0; j<num_deadzones; j++) {
            unsigned int(16) deadzone_left_horizontal_offset;
            unsigned int(16) deadzone_top_vertical_offset;
            unsigned int(16) deadzone_width;
            unsigned int(16) deadzone_height;
        }
    }
}
```

6.3.2 Semantics

`num_circular_images` specifies the number of circular images in the cropped output picture of each coded picture this structure applies to. Typically, the value is equal to 2, but other non-zero values are also possible.

`entrance_pupil_flag` equal to 1 specifies the entrance pupil coefficients are present. `entrance_pupil_flag` equal to 0 specifies that the entrance pupil coefficients are not present. When `container_box_version` is equal to 0, `entrance_pupil_flag` shall be equal to 0.

`flip_info_flag` equal to 1 specifies that `image flipping` region information is present. `flip_info_flag` equal to 0 specifies that image flipping information is not present.

`camera_intrinsic_flag` equal to 1 specifies that image scale and lens shading compensation camera intrinsic parameters are present. `camera_intrinsic_flag` equal to 0 specifies that image scale and lens shading compensation camera intrinsic parameters are not present.

`local_fov_flag` equal to 1 specifies that the local field of view parameters are present. `local_fov_flag` equal to 0 specifies that the local field of view parameters are not present.

`deadzone_flag` equal to 1 specifies that deadone related parameters are present. `deadzone_flag` equal to 0 specifies that deadzone parameters are not present.

`lens_projection_type` indicates the type of lens projection model corresponding to the circular image. `lens_projection_type` equal to 0 indicates the equidistant projection model. `lens_projection_type` equal to 1 indicates the perspective projection model. `lens_projection_type` equal to 2 indicates the stereographic projection model. `lens_projection_type` equal to 3 indicates the sine-law projection model. `lens_projection_type` equal to 4 indicates the equi-solid angle projection model. Values for `lens_projection_type` in the range of 5 to 15, inclusive, are reserved.

num_ep_coeffs specifies the number of entrance pupil coefficients for the circular image. The value of num_ep_coeffs shall be equal to 3, 5, 7, or 9, and other values of num_ep_coeffs are reserved.

The instances of ep_coeff are entrance pupil distortion parameters that define the curve function that corrects the incidence ray of any object which points to the optical center C_W as shown in Figure 11. These entrance pupil distortion parameters are used to correct the camera_centre_offset_z (O_Z) to its real optical center (C_W) in the external camera coordinates.

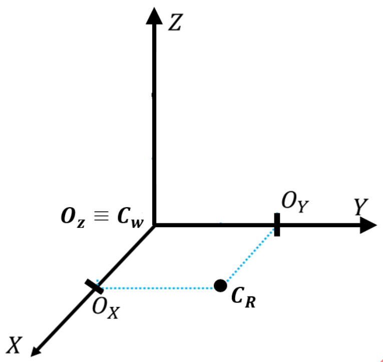


Figure 11 — Illustration of the entrance pupil rectification on the Z-axis

The instances of ep_coeff are fixed-point 8.24 polynomial coefficient values q_j. For the first given ep_coeff the value of j is equal to 3, and j is incremented by 1 for each subsequent instance of ep_coeff. The values of q_j specify the entrance pupil equation:

$$EP(\theta) = \sum_{j=3}^{M_i-1} q_j * r^j \tag{10}$$

where M_i is represented by num_ep_coeffs + 3.

When instances of ep_coeff are present, the angle (θ) of a sphere coordinate from the normal vector of a nominal imaging plane that passes through the centre of the sphere coordinate system for the active area of the circular image, where the angle is in units of radians, is specified using the following polynomial equation:

$$\theta = \sum_{j=0}^{N_i-1} p_j * r^j + \mathrm{EP}(\theta) \tag{11}$$

where p_j and N_i are represented by polynomial_coeff and num_polynomial_coeffs, respectively, and r is the normalized distance of a luma sample (x_p) from the centre of the circular image.

image_flip specifies whether and how the image has been flipped and thus a reverse flipping operation needs to be applied. The value 0 indicates that the image has not been flipped. The value 1 indicates that the image has been vertically flipped. The value 2 indicates that the image has been horizontally flipped. The value 3 indicates that the image has been both vertically and horizontally flipped.

image_scale_axis_angle, image_scale_x, and image_scale_y are three fixed-point 16.16 values that specify whether and how the circular image has been scaled along an axis. They are used to take into account the natural error in the camera-mirror setting. The axis is defined by a single angle as indicated by the value of image_scale_axis_angle, in degrees. An angle of 0 degrees means a horizontal vector is perfectly horizontal and a vertical vector is perfectly vertical. The values of image_scale_x and image_scale_y indicate the scaling. They may also be called as affine parameters that satisfy the following equation:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} c & d \\ e & 1 \end{bmatrix} \begin{bmatrix} u_N \\ v_N \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{12}$$

The equation relates the actual pixel coordinates (u, v) to the ideal image coordinates (u_N, v_N). c_x and c_y represent image_centre_x, image_centre_y respectively. c, d, and e represent image_scale_x, image_scale_axis_angle, and image_scale_y respectively.

num_polynomial_coefs_lsc specifies the number of polynomial coefficients of the lens shading compensation (LSC) parameters for the circular image.

polynomial_coef_k_lsc_r, polynomial_coef_k_lsc_g, and polynomial_coef_k_lsc_b are 8.24 fixed-point values that specify the LSC parameters to compensate the shading artifact of the fisheye lense that reduces the color in the radial direction. The compensating weight (w) to be multiplied to the original color is approximated as a curve function using the following polynomial equations where r is the normalized radius as specified in subclause 6.2.2:

If entrance_pupil_flag is equal to 0, the compensating weight (w) is specified as follows:

$$\mathrm{w} = \sum_{i=1}^{N} p_{i-1} * r^{i-1} \tag{13}$$

Otherwise, the compensating weight (w) is specified as follows:

$$\mathrm{w} = \sum_{i=1}^{N} p_{i-1} * r^{i-1} + EP(\theta) \tag{14}$$

The weighing factors for red, green, and blue may be calculated separately when p is represented by polynomial_coef_k_lsc_r, polynomial_coef_k_lsc_g, and polynomial_coef_k_lsc_b, respectively, and r is the corresponding radius from the image centre after normalization by the full_radius. N is represented by num_polynomial_coefs_lsc. Figure 12 shows an example of an input fisheye video picture and the output after applying lens shading compensation.

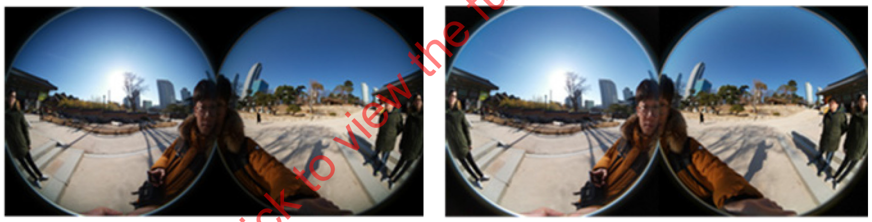


Figure 12 — Example of a pair of output images (on the right) resulting from lens shading compensation from a pair of circular input images (on the left)

num_angle_for_displaying_fov specifies the number of angles that define the displayed and overlapped regions. According to the value of num_angle_for_displaying_fov, multiple values of displayed_fov and overlapped_fov are defined with equal intervals, which start at 12 o'clock and increase clockwise.

displayed_fov specifies the field of view of the part of the circular image that is recommended to be used for displaying without blending that involves adjacent circular images.

overlapped_fov specifies the field of view of the part of the circular image that may be overlapped on a sphere by adjacent circular images and that is recommended to be used for displaying either as such or by blending with adjacent circular images.

The values of displayed_fov and overlapped_fov shall be less than or equal to the value of field_of_view.

NOTE The value of field_of_view is determined by the physical property of each fisheye lens, while the values of displayed_fov and overlapped_fov are determined by the configuration of multiple fisheye lenses. For example, when the value of num_circular_images is equal to 2 and two lenses are symmetrically located, the value of displayed_fov and overlapped_fov can be set as 180 and 190 respectively, by default. However, the value can be changed depending on the configuration of the lens and the characteristics of the contents. For example, if the stitching quality with the displayed_fov values (left camera = 170 and right camera = 190) and the overlapped_fov values (left camera = 185 and right camera = 190) is better than the quality with the default values (180 and 190) or if the physical configuration of cameras is asymmetric, then the unequal displayed_fov and overlapped_fov values can be taken. In addition, when it comes to multiple (N>2) fisheye images, a single displayed_fov value cannot specify the exact area of each fisheye image. As shown in Figure 13 and Figure 14, displayed_fov (red-coloured) varies according to the direction. In order to manipulate multiple (N>2) fisheye images, num_angle_for_displaying_fov is introduced. For example, if this value is equal to 12, then the fisheye image is divided into 12 sectors where each sector angle is 30 degrees.

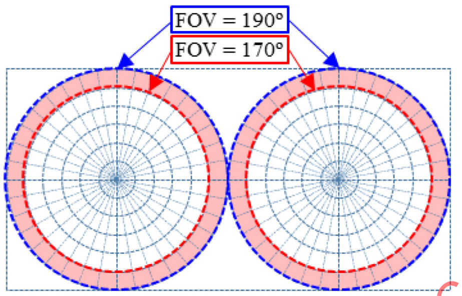


Figure 13 — Displayed FOV for two fisheye images (=170 or 190)

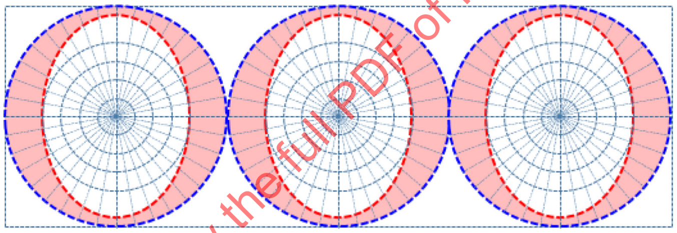


Figure 14 — Displayed FOV and overlapped FOV for multiple (N>2) fisheye images

num_local_fov_region specifies the number of local fitting regions that have different fields of view.

start_radius (r_s), end_radius (r_e), start_angle (a_s), and end_angle (a_e) specify the region for local fitting/warping to change the actual field of view for displaying locally. start_radius and end_radius are fixed-point 16.16 values that specify the minimum and maximum radius values. start_angle and end_angle specify the minimum and maximum angle values that start at 12 o'clock and increase clockwise, in units of 2^{-16} degrees. start_angle and end_angle shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive. Figure 15 illustrates an example of local FOV parameters.

radius_delta (r_d) is a fixed-point 16.16 value that specifies the delta radius value for representing a different field of view for each radius.

angle_delta (a_d) specifies the delta angle value, in units of 2^{-16} degrees, for representing a different field of view for each angle.

local_fov_weight (W) is a 8.24 fixed point value which specifies the weighting value for the field of view of the position specified by start_radius, end_radius, start_angle, end_angle, the angle index i and the radius index j. The positive value of local_fov_weight specifies the expansion of field of view, while the negative value specifies the contraction of field of view. Figure 16 shows an example result of applying local FOV parameters with a set of weighting values.

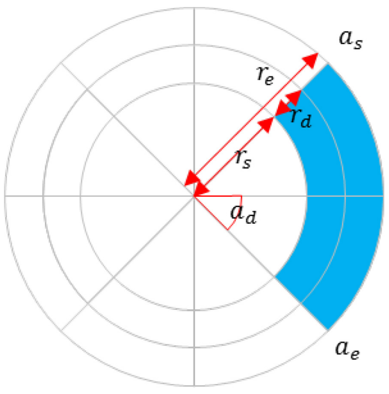


Figure 15 — Illustration of parameters regarding local FOV

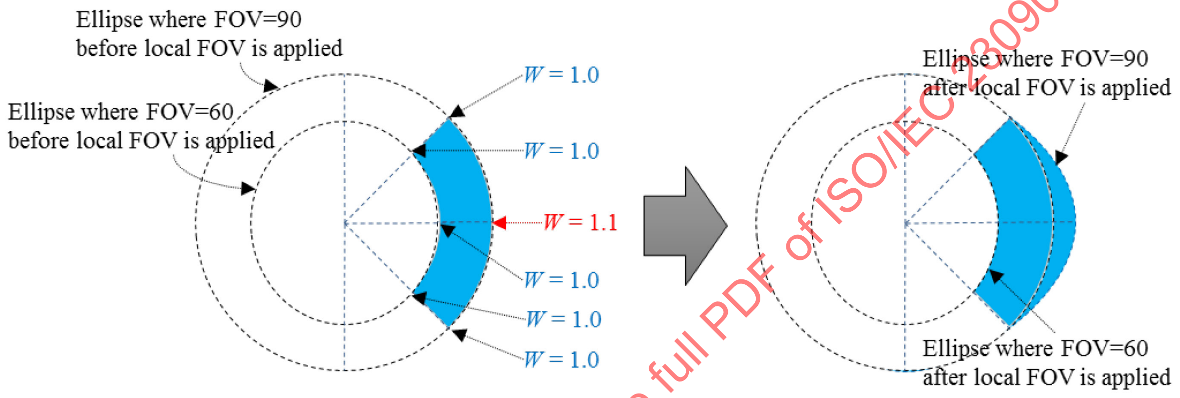


Figure 16 — Example of local FOV with a_s, a_e, and a_d equal to 45°, 135°, and 45°, respectively, and r_s, r_e, and r_d equal to 0.5, 1.0, and 0.5, respectively

`num_deadzones` is an integer that specifies the number of dead zones in each coded picture this structure applies to.

`deadzone_left_horizontal_offset`, `deadzone_top_vertical_offset`, `deadzone_width`, and `deadzone_height` are integer values that specify the position and size of the deadzone rectangular area in which the pixels are not usable. `deadzone_left_horizontal_offset` and `deadzone_top_vertical_offset` specify the horizontal and vertical coordinates, respectively, in luma samples, of the upper left corner of the deadzone in the coded picture. `deadzone_width` and `deadzone_height` specify the width and height, respectively, in luma samples, of the deadzone. To save bits for representing the video, all pixels within a deadzone should be set to the same pixel value, e.g. all black.

7 Omnidirectional media storage and metadata signalling in the ISOBMFF

7.1 Generic extensions to the ISOBMFF

NOTE Content found in ISO/IEC 23090-2:2019, subclauses 7.1.1, 7.1.2, 7.1.3, 7.1.4, 7.1.5.1, 7.1.6, 7.1.7, 7.1.8 and 7.1.9 (the first edition of this document) has been integrated into ISO/IEC 14496-12 and subclause 7.1 has been renumbered.

7.1.1 Indication of a track not intended to be presented alone

Bit 4 of the flags (with bit 0 being the least significant bit) of the TrackHeaderBox is used to indicate whether a track is not intended to be presented alone, e.g. due to that the track represents only a small portion of a videos scene. The semantics of the flag is specified as follows:

> track_not_intended_for_presentation_alone: Indicates that the track is not intended to be presented alone without other tracks. Flag value is 0x000010. The flag not being set (i.e. flags & 0x000010 is equal to 0) indicates that the track may or may not be intended to be presented alone without other tracks.

The use of the track_not_intended_for_presentation_alone flag is deprecated.

7.1.2 Association of timed metadata tracks with media tracks or track groups

7.1.2.1 Association with media tracks by the 'cdtg' track reference

A timed metadata track containing a 'cdtg' track reference describes the referenced media tracks and track groups collectively. The 'cdtg' track reference shall only be present in timed metadata tracks.

NOTE A timed metadata track containing 'cdsc' track reference to a track_group_id value describes each track in the track group individually.

When a timed metadata track contains a 'cdtg' track reference to a track group of type '2dsr', the timed metadata track describes the composition picture.

7.1.3 Clarifications on the stereo video box

This document uses StereoVideoBox as specified in ISO/IEC 14496-12 with the following additional specifications:

— When the value of stereo_indication_type indicates the temporal interleaving frame packing arrangement, each sync sample and each SAP sample with SAP_type in the range of 1 to 3, inclusive, indicated by the stream access point sample group represent constituent picture 0 followed, in composition time order, by samples representing constituent pictures 1 and 0 in an alternating manner up to but excluding the next sync sample or SAP sample with SAP_type in the range of 1 to 3, inclusive, indicated by the stream access point sample group.

7.1.4 Generic sub-picture track grouping extensions

7.1.4.1 Updated semantics of track_group_type

The semantics of track_group_type of the TrackReferenceBox is changed from

> track_group_type indicates the grouping_type and shall be set to one of the following values, or a value registered, or a value from a derived specification or registration:
>
> 'msrc' indicates that this track belongs to a multi-source presentation. Specified in 8.3.4.4.1.
>
> 'ster' indicates that this track is either the left or right view of a stereo pair suitable for playback on a stereoscopic display. Specified in 8.3.4.4.2.
>
> The pair of track_group_id and track_group_type identifies a track group within the file. The tracks that contain a particular TrackGroupTypeBox having the same value of track_group_id and track_group_type belong to the same track group.

to

track_group_type indicates the grouping_type and shall be set to one of the following values, or a value registered, or a value from a derived specification or registration:

'msrc' indicates that this track belongs to a multi-source presentation. Specified in ISO/IEC 14496-12:2020, subclause 8.3.4.4.1.

'ster' indicates that this track is either the left or right view of a stereo pair suitable for playback on a stereoscopic display. Specified in ISO/IEC 14496-12:2020, subclause 8.3.4.4.2.

'2dsr' indicates that this track belongs to a group of tracks with two dimensional spatial relationships (e.g. corresponding to spatial parts of a video source). Specified in subclause 7.1.4.2 of this document.

The pair of track_group_id and track_group_type identifies a track group within the file. The tracks that contain a particular TrackGroupTypeBox having the same value of track_group_id and track_group_type belong to the same track group.

7.1.4.2 Two dimensional spatial relationship

7.1.4.2.1 Definition

A SpatialRelationship2DDescriptionBox TrackGroupTypeBox indicates that this track belongs to a group of tracks with 2D spatial relationships (e.g. corresponding to planar spatial parts of a video source). A SpatialRelationship2DDescriptionBox TrackGroupTypeBox with a given track_group_id implicitly defines a coordinate system with an arbitrary origin (0,0) and a maximum size defined by total_width and total_height; the x-axis is oriented from left to right and the y-axis from top to bottom. The tracks that have the same value of source_id within a SpatialRelationship2DDescriptionBox TrackGroupTypeBox are mapped as being originated from the same source and their associated coordinate systems share the same origin (0,0) and the orientation of their axes. For example, a very high resolution video can have been split into sub-picture tracks. Each sub-picture track then conveys its position and sizes in the source video.

Tracks in the same track group shall declare the same source_id, total_width, and total_height. Track groups with different track_group_id values and the same source_id represent the same source content, possibly at different resolutions (i.e. with different values of total_width or total_height).

NOTE A source can be represented by different such track groups (for instance when the same source is available at different resolutions). Each of these track groups is identified by its own identifier track_group_id. Since all of these track groups originate from the same source, they share the same source_id.

The SubPictureRegionBox is optional and either:

a) is present in the SpatialRelationship2DDescriptionBox and there shall be no associated SpatialRelationship2DGroupEntry in the associated track (this track has a constant, static, size and position); or

b) is not present in the SpatialRelationship2DDescriptionBox and there shall be one or more associated SpatialRelationship2DGroupEntry(s) in the associated track (this track possibly has a dynamic size or position).

When constructing the composition picture as specified below, gaps between sub-pictures carried in different sub-picture tracks are allowed, and overlaps between sub-pictures carried in different sub-picture tracks with different values of the layer field in TrackHeaderBox are allowed. However, overlaps between sub-pictures carried in different sub-picture tracks with the same value of the layer field in TrackHeaderBox are not allowed.

The spatial relationship is restricted according to the chroma sub-sampling format of the associated track; `total_width` and `total_height`, and `object_x`, `object_y`, `object_width` and `object_height`, shall all select an integer number of samples for all planes. In effect this means that:

— when the format is 4:4:4, there is no restriction;

— when the format is 4:2:2 the `total_width`, `object_x` and `object_width` shall be even numbers;

— when the format is 4:2:0 all of these fields shall be even numbers.

The composition picture is reconstructed as follows, with values of `object_x`, `object_y`, `object_width`, and `object_height` obtained from `SubPictureRegionBox` if present or otherwise from the `SpatialRelationship2DGroupEntry` applying to the sample:

a) Out of all tracks belonging to the same `'2dsr'` track group, form them into subgroups such that each subgroup contains tracks in the same alternate group; then select exactly one track from each of those subgroups.

b) For each composition-time aligned sample of each of the selected tracks, the following applies, in the front-to-back ordering (`layer`) indicated in the `TrackHeaderBox` of the picked tracks:

For each value of i in the range of 0 to `object_width` − 1, inclusive, and for each value of j in the range of 0 to `object_height` − 1, inclusive, the pixel value of the composition picture at pixel position ((i + `object_x`) % `total_width`, (j + `object_y`) % `total_height`) is set equal to the pixel value of the sample of this track at pixel position (i, j).

7.1.4.2.2 Syntax

```
aligned(8) class SpatialRelationship2DSourceBox
      extends FullBox('2dss', 0, 0) {
      unsigned int(32) total_width;
      unsigned int(32) total_height;
      unsigned int(32) source_id;
}

aligned(8) class SubPictureRegionBox extends FullBox('sprg',0,0) {
      unsigned int(16) object_x;
      unsigned int(16) object_y;
      unsigned int(16) object_width;
      unsigned int(16) object_height;
      bit(14) reserved = 0;
      unsigned int(1) track_not_alone_flag;
      unsigned int(1) track_not_mergeable_flag;
}

aligned(8) class SpatialRelationship2DDescriptionBox extends
TrackGroupTypeBox('2dsr') {
      // track_group_id is inherited from TrackGroupTypeBox;
      SpatialRelationship2DSourceBox();    // mandatory, shall be first
      SubPictureRegionBox();               // optional
}
```

7.1.4.2.3 Semantics

`total_width` specifies, in pixel units, the maximum width in the coordinate system of the `SpatialRelationship2DDescriptionBox` track group. The value of `total_width` shall be the same in all instances of `SpatialRelationship2DDescriptionBox` with the same value of `track_group_id`.

total_height specifies, in pixel units, the maximum height in the coordinate system of the SpatialRelationship2DDescriptionBox track group. The value of total_height shall be the same in all instances of SpatialRelationship2DDescriptionBox with the same value of track_group_id.

source_id parameter provides a unique identifier for the source. It implicitly defines a coordinate system associated to this source.

object_x specifies the horizontal position of the top-left corner of the samples in this track within the coordinate system specified by this spatial relationship track group. The position value is the value prior to applying the implicit resampling caused by the track width and height, if any, in the range of 0 to total_width − 1, inclusive, where total_width is included in this SpatialRelationship2DDescriptionBox.

object_y specifies the vertical position of the top-left corner of the samples in this track within the coordinate system specified by this spatial relationship track group. The position value is the value prior to applying the implicit resampling caused by the track width and height, if any, in the range of 0 to total_height − 1, inclusive, where total_height is included in this SpatialRelationship2DDescriptionBox.

object_width specifies the width of the samples in this track within the coordinate system specified by this spatial relationship track group. The width value is the value prior to applying the implicit resampling caused by the track width and height, if any, in the range of 1 to total_width, inclusive.

object_height specifies the height of the samples in this track within the coordinate system specified by this spatial relationship track group. The height value is the value prior to applying the implicit resampling caused by the track width and height, if any, in the range of 1 to total_height, inclusive.

track_not_alone_flag equal to 1 indicates that the current sub-picture track is not intended to be presented alone without at least one other sub-picture track belonging to the same track group of grouping type '2dsr'. The value 0 indicates that the current sub-picture track may or may not be intended to be presented alone without at least one other sub-picture track belonging to the same track group of grouping type '2dsr'.

track_not_mergeable_flag equal to 1 indicates that the video bitstream carried in the current sub-picture track cannot be merged with the video bitstream carried in any other sub-picture tracks belonging to the same track group of grouping type '2dsr', to generate a single video bitstream without decoding mismatch by rewriting only header data of the bitstreams, where a decoding mismatch refers to the value of any pixel when decoding the video bitstream in the current track is not identical to the value of the same pixel when decoding the merged single video bitstream. An example of such bitstream merging is the reconstruction of an HEVC bitstream as specified in subclause 10.1.3.4 when the untransformed sample entry type of the track with the given track_ID is equal to 'hvc2'. track_not_mergeable_flag equal to 0 indicates that the video bitstream carried in the current sub-picture track can be merged with the video bitstream carried in at least one other sub-picture track belonging to the same track group of grouping type '2dsr' to generate such a single video bitstream in such a manner as described above.

NOTE When HEVC (i.e. Rec. ITU-T H.265 | ISO/IEC 23008-2) is the video codec used for encoding of the bitstreams carried in the sub-picture tracks, track_not_mergeable_flag equal to 0 means that the HEVC bitstream carried in the current sub-picture track contains and only contains one or more MCTSs that can be indicated by a temporal MCTSs SEI message as specified in HEVC version 5 published by the ITU-T in Feburary 2018, or a later version of HEVC.

7.1.4.3 Spatial relationship 2D sample group

7.1.4.3.1 Definition

The '2dsr' grouping_type for sample grouping declares the positions and sizes of the samples from a sub-picture track in a spatial relationship track group. Version 1 of the SampleToGroupBox shall be used when grouping_type is equal to '2dsr'. The value of grouping_type_parameter shall be equal to track_group_id of the corresponding spatial relationship track group.

There are restrictions both on the presence of this sample grouping, and on the values of the fields; see subclause 7.1.4.2.1.

7.1.4.3.2 Syntax

```
class SpatialRelationship2DGroupEntry extends VisualSampleGroupEntry('2dsr') {
      unsigned int(16) object_x;
      unsigned int(16) object_y;
      unsigned int(16) object_width;
      unsigned int(16) object_height;
}
```

7.1.4.3.3 Semantics

object_x specifies the horizontal position of the top-left corner of the samples in this group within the coordinate system specified by the corresponding spatial relationship track group. The position value is the value prior to applying the implicit resampling caused by the track width and height, if any, in the range of 0 to total_width − 1, inclusive, where total_width is included in the corresponding SpatialRelationship2DDescriptionBox.

object_y specifies the vertical position of the top-left corner of the samples in this group within the coordinate system specified by the corresponding spatial relationship track group. The position value is the value prior to applying the implicit resampling caused by the track width and height, if any, in the range of 0 to total_height − 1, inclusive, where total_height is included in the corresponding SpatialRelationship2DDescriptionBox.

object_width specifies the width of the samples in this group within the coordinate system specified by the corresponding spatial relationship track group. The width value is the value prior to applying the implicit resampling caused by the track width and height, if any, in the range of 1 to total_width, inclusive.

object_height specifies the height of the samples in this group within the coordinate system specified by the corresponding spatial relationship track group. The height value is the value prior to applying the implicit resampling caused by the track width and height, if any, in the range of 1 to total_height, inclusive.

7.1.5 Track reference indicating a track providing shadow sync samples

In this subclause a track containing an 'shsc' track reference is called a shadow sync sample track, and the tracks pointed to by the 'shsc' track reference are called main tracks.

The shadow sync sample track provides an optional set of sync samples that can be used when seeking to a position or for similar operations performed to any of the associated main tracks.

When an 'shsc' track reference is present, the following constraints shall be obeyed:

— All samples of the shadow sync sample track shall be sync samples.

— Each main track shall have a sample that is aligned in decoding time with each sample of the shadow sync sample track.

— A concatenation of the following samples in the following order shall conform to the sample entry of the main track:

  — Any selected sample of the shadow sync sample track, with the sample duration of the sample of the main track that is aligned in decoding time with the selected sample of the shadow sync sample track.

  — Samples of the main track following the sample of the main track that is aligned in decoding time with the selected sample of the shadow sync sample track.

An 'shsc' track reference indicates that the decoded samples resulting from the concatenation specified above have acceptable quality for playback.

NOTE 1 The samples in the main track that are aligned in decoding time with the samples in the shadow sync sample track are "switchable" samples that are constrained so that no samples preceding a "switchable" sample in decoding order are used as a prediction reference for any sample following the "switchable" sample in decoding order.

NOTE 2 Figure 17 presents an example of a shadow sync sample track and the associated main track. The concatenation of any selected sample of the shadow sync sample track and the samples of the main track following the "switchable" sample that is aligned in decoding time with the selected samples conforms to the sample entry of the main track. This implies that the bitstream contained in the concatenation of the samples conforms to the respective media coding specification.

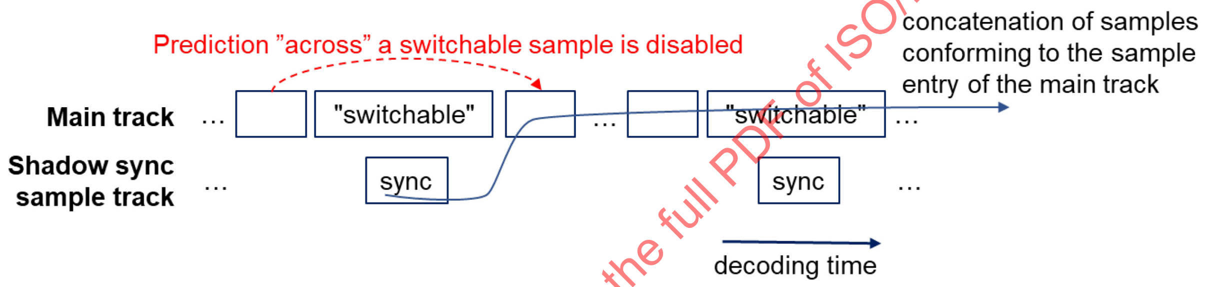

Prediction "across" a switchable sample is disabled
concatenation of samples conforming to the sample entry of the main track
Main track
"switchable"
"switchable"
Shadow sync sample track
sync
sync
decoding time

Figure 17 — Example of a shadow sync sample track and the associated main track

7.1.6 Media offset box

7.1.6.1 Definition

Box Type:	'meof'
Container:	File
Mandatory:	No
Quantity:	Zero or more

MediaOffsetBox provides media offsets of the media segments that are referenced by the preceding SegmentIndexBox. Each media offset specifies the byte offset of the first media data box (either MediaDataBox or IdentifiedMediaDataBox) that contains media data for the media segment.

There shall be 0 or 1 MediaOffsetBoxes per each SegmentIndexBox. A MediaOffsetBox, if any, should be the next box after the associated SegmentIndexBox and the SubsegmentIndexBox, if any, that immediately follows the SegmentIndexBox. A MediaOffsetBox is associated with the preceding SegmentIndexBox.

7.1.6.2 Syntax

```
aligned(8) class MediaOffsetBox() extends FullBox ('meof', 0, 0) {
   unsigned int(16) media_ref_count;
   for(i=1; i <= media_ref_count; i++) {
      unsigned int(32) media_data_offset[i];
      unsigned int(31) media_data_size[i];
      unsigned int(1) sap_size_flag[i];
      if (sap_size_flag[i]) {
         unsigned int(31) sap_size[i];
         unsigned int(1) sap_id_flag[i];
         if (sap_id_flag[i])
            unsigned int(32) sap_id[i];
      }
   }
}
```

7.1.6.3 Semantics

media_ref_count specifies the number of referenced media data boxes. The value of media_ref_count shall be equal to the number of entries with reference_type equal to 0 in the preceding SegmentIndexBox. The referenced media data box (i.e. the referenced MediaDataBox or IdentifiedMediaBox) for loop entry i in MediaOffsetBox corresponds to the i-th entry with reference_type equal to 0 in the preceding SegmentIndexBox. In other words, loop entry i in MediaOffsetBox is a 1-based index to the list of entries that have reference_type equal to 0 in the preceding SegmentIndexBox in the order the entries appear in the SegmentIndexBox.

media_data_offset[i] specifies the offset to the start of the referenced MediaDataBox or the IdentifiedMediaDataBox of a subsegment. The value of media_data_offset[i] is relative to the start of the segment containing the referenced MediaDataBox or the IdentifiedMediaDataBox.

media_data_size[i] specifies byte count of the referenced MediaDataBox or IdentifiedMediaDataBox of a subsegment.

sap_size_flag[i] equal to 1 specifies that the sap_size[i] and reserved syntax elements are present for the same value of i. sap_size_flag[i] equal to 0 specifies that sap_size[i] and reserved syntax elements are not present for the same value of i. When starts_with_SAP for a subsegment is equal to 0, sap_size_flag[i] shall be equal to 0 for the same subsegment. When sap_size_flag[i] is equal to 1, the SAP sample shall start from the beginning of the box payload of MediaDataBox or from the byte following imda_identifier in IdentifiedMediaDataBox.

sap_size[i] specifies the byte count of the SAP sample that starts a subsegment.

sap_id_flag[i] equal to 1 specifies that sap_id[i] is present. sap_id_flag[i] equal to 0 specifies that sap_id[i] is not present.

sap_id[i] identifies a SAP sample. Samples with the same value of sap_id[i] are identical. When a subsegment contains an initial SAP sample with a sap_id[i] value that is already available in a player, the player may avoid downloading the SAP sample and use the already available identical copy of the SAP sample instead.

7.2 Generic extensions to ISO/IEC 14496-15

NOTE Content found in ISO/IEC 23090-2:2019, subclauses 7.2.1 and 7.2.2 (the first edition of this document) has been integrated into ISO/IEC 14496-15 and subclause 7.2 has been renumbered.

7.2.1 Containing of `SpatialRelationship2DDescriptionBox` for HEVC tile base track and HEVC tile tracks

When an HEVC video bitstream is carried in a set of tile tracks and the associated tile base track, as specified in ISO/IEC 14496-15, and the bitstream represents a sub-picture indicated by a 2D spatial relationship track group, only the tile base track contains the `SpatialRelationship2DDescriptionBox`.

7.3 OMAF-specific extensions to the ISOBMFF

7.3.1 Sync samples in timed metadata tracks

For timed metadata tracks specified in this document, any sample in a timed metadata track is allowed to be marked as a sync sample. For a particular sample in a timed metadata track, if at least one of the media samples in the referenced media tracks having the same decoding time is a sync sample, the particular sample shall be marked as a sync sample, otherwise, the particular sample may or may not be marked as a sync sample.

7.4 OMAF-specific extensions to ISO/IEC 14496-15

7.4.1 Coverage information box in an HEVC tile base track

When `CoverageInformationBox` is included in `ProjectedOmniVideoBox` of a tile base track, it provides information on the area on the sphere covered by the content that is represented by all the tile tracks associated with the tile base track.

7.5 Structures and semantics that are common for video tracks and image items

7.5.1 Semantics of sample locations within a decoded picture

7.5.1.1 Relation of decoded pictures to global coordinate axes

Figure 18 illustrates the conversions from a spherical picture to a packed picture that can be used in content authoring and the corresponding conversions from a packed picture to a spherical picture to be rendered that can be used in an OMAF player. The example in this subclause is described for a packed picture that appears in a projected omnidirectional video track. Similar description can be derived for an image item.

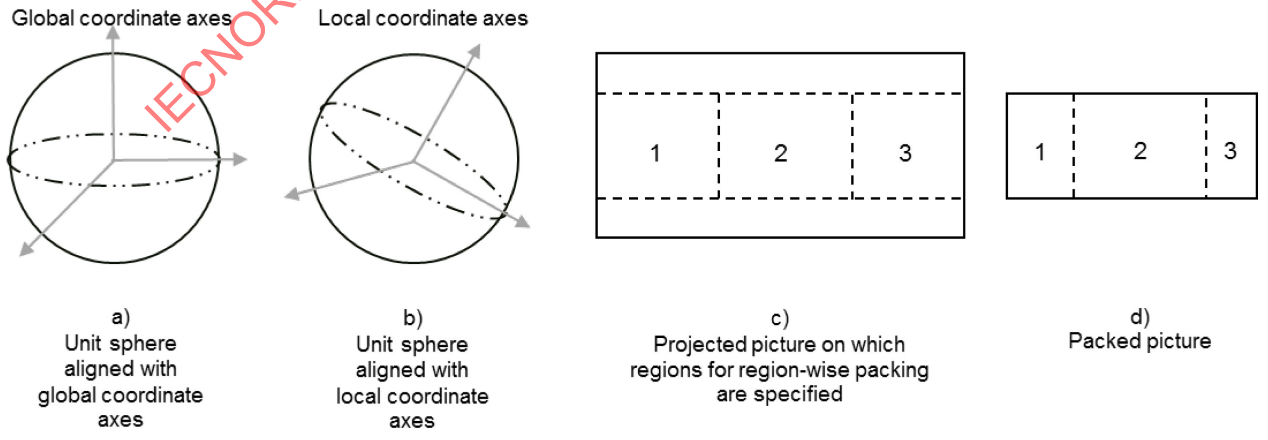

Global coordinate axes
Local coordinate axes
1
2
3
1
2
3
a)
Unit sphere aligned with global coordinate axes
b)
Unit sphere aligned with local coordinate axes
c)
Projected picture on which regions for region-wise packing are specified
d)
Packed picture

Figure 18 — Example of processing stages to derive a packed picture from a spherical image or vice versa

The content authoring can include the following ordered steps:

- The source images provided as input are stitched to generate a sphere picture on the unit sphere per the global coordinate axes as indicated in Figure 18a.
- The unit sphere is then rotated relative to the global coordinate axes, as indicated in Figure 18b. The amount of rotation to convert from the local coordinate axes to the global coordinate axes is specified by the rotation angles indicated in the `RotationBox`. The local coordinate axes of the unit sphere are the axes of the coordinate system that has been rotated. The absence of `RotationBox` indicates that the local coordinate axes are the same as the global coordinate axes.
- As illustrated in Figure 18c, the spherical picture on the rotated unit sphere is then converted to a two-dimensional projected picture, for example using the equirectangular projection specified in subclause 5.2.1. When spatial packing of stereoscopic content is applied, two spherical pictures for the two views are converted to two constituent pictures, after which frame packing is applied to pack the two constituent pictures to one projected picture.
- Rectangular region-wise packing can be applied to obtain a packed picture from the projected picture. One example of packing is depicted in Figure 18c and Figure 18d. The dashed rectangles in Figure 18c indicate the projected regions on a projected picture, and the respective areas in Figure 18d indicate the corresponding packed regions. In this example, projected regions 1 and 3 are horizontally downsampled, while projected region 2 is kept at its original resolution.

`CoverageInformationBox` can be used to indicate which part of the sphere is covered by the packed picture.

In order to map sample locations of a packed picture (such as that in Figure 18d) to a unit sphere used in rendering (Figure 18a), the OMAF player can perform the following ordered steps:

- A packed picture, such as that in Figure 18d, is obtained as a result of decoding a picture from a video track or an image item.
- If needed, chroma sample arrays of the packed picture are upsampled to the resolution of the luma sample array of the packed picture, and colour space conversion can also be performed.
- If region-wise packing is indicated, the sample locations of the packed picture are converted to sample locations of the respective projected picture, such as that in Figure 18c, as specified in subclause 5.4.2. Otherwise, the projected picture is identical to the packed picture.
- If spatial frame packing of the projected picture is indicated, the sample locations of the projected picture are converted to sample locations of the respective constituent picture of the projected picture, as specified in subclause 7.5.1.3. Otherwise, the constituent picture of the projected picture is identical to the projected picture.
- The sample locations of a constituent picture the projected picture are converted to sphere coordinates that are relative to local coordinate axes, as specified for the omnidirectional projection format being used in subclause 5.2. The resulting sample locations correspond to a sphere picture depicted in Figure 18b.
- If rotation is indicated, the sphere coordinates relative to the local coordinate axes are converted to sphere coordinates relative to the global coordinate axes as specified in subclause 5.3. Otherwise, the global coordinate axes are identical to the local coordinate axes.

The overall process for mapping of luma sample locations within a decoded picture to sphere coordinates relative to the global coordinate axes is specified in subclause 7.5.1.2.

7.5.1.2 Mapping of luma sample locations within a decoded picture to sphere coordinates relative to the global coordinate axes

This subclause specifies the semantics of luma sample locations within a decoded picture to sphere coordinates relative to the global coordinate axes. The decoded picture may be of any of the following:

- For video, the decoded picture is the decoding output resulting from a sample of the video track.
- For an image item, the decoded picture is a reconstructed image of the image item.

offsetX is set equal to 0.5 and offsetY is set equal to 0.5.

If RegionWisePackingFlag is equal to 1, the following applies for each packed region n in the range of 0 to NumRegions − 1, inclusive:

- For each sample location (xPackedPicture, yPackedPicture) belonging to the n-th packed region with PackingType[n] equal to 0 (i.e. with rectangular region-wise packing), the following applies:
  - The corresponding sample location (xProjPicture, yProjPicture) of the projected picture is derived as follows:
    - x is set equal to xPackedPicture − PackedRegLeft[n].
    - y is set equal to yPackedPicture − PackedRegTop[n].
    - Subclause 5.4.2 is invoked with x, y, PackedRegWidth[n], PackedRegHeight[n], ProjRegWidth[n], ProjRegHeight[n], TransformType[n], offsetX, and offsetY as inputs, and the output is assigned to sample location (hPos, vPos).
    - xProjPicture is set equal to ProjRegLeft[n] + hPos.
    - When SideBySideFlag is equal to 0, and when xProjPicture is greater than or equal to `proj_picture_width`, xProjPicture is set equal to xProjPicture − `proj_picture_width`.
    - When SideBySideFlag is equal to 1, the following applies:
      - When ProjRegLeft[n] is less than `proj_picture_width` / 2 and xProjPicture is greater than or equal to `proj_picture_width` / 2, xProjPicture is set equal to xProjPicture − `proj_picture_width` / 2.
      - When ProjRegLeft[n] is greater than or equal to `proj_picture_width` / 2 and xProjPicture is greater than or equal to `proj_picture_width`, xProjPicture is set equal to xProjPicture − `proj_picture_width` / 2.
    - yProjPicture is set equal to ProjRegTop[n] + vPos.
  - Subclause 7.5.1.3 is invoked with xProjPicture, yProjPicture, ConstituentPicWidth, and ConstituentPicHeight as inputs, and the outputs indicating the sphere coordinates and the constituent frame index (for frame-packed stereoscopic video) for the luma sample location (xPackedPicture, yPackedPicture) belonging to the n-th packed region in the decoded picture.

Otherwise (RegionWisePackingFlag is equal to 0), the following applies for each sample location (x, y) within the decoded picture:

- xProjPicture is set equal to x + offsetX.
- yProjPicture is set equal to y + offsetY.

— Subclause 7.5.1.3 is invoked with xProjPicture, yProjPicture, ConstituentPicWidth, and ConstituentPicHeight as inputs, and the outputs indicating the sphere coordinates and the constituent frame index (for frame-packed stereoscopic video) for the sample location (x, y) within the decoded picture.

7.5.1.3 Conversion from a sample location in a projected picture to sphere coordinates relative to the global coordinate axes

Inputs to this process are

— the centre point of a sample location (xProjPicture, yProjPicture) within a projected picture, where xProjPicture and yProjPicture are in relative projected picture sample units and may have non-integer real values, and

— pictureWidth and pictureHeight, which are the width and height, respectively, of a monoscopic projected luma picture, in relative projected picture sample units.

NOTE 1 The projected picture for which the sample location (xProjPicture, yProjPicture) is given as input can be a spatially frame-packed picture.

Outputs of this process are:

— sphere coordinates (azimuthGlobal, elevationGlobal), in units of degrees relative to the global coordinate axes, and

— when SpatiallyPackedStereoFlag is equal to 1, the index of the constituent picture (constituentPicture) equal to 0 or 1.

NOTE 2 When the temporal interleaving packing arrangement is in use, the projected picture is associated with the left view or right view as specified for `StereoVideoBox` in ISO/IEC 14496-12.

The outputs are derived with the following ordered steps:

— If xProjPicture is greater than or equal to pictureWidth or yProjPicture is greater than or equal to pictureHeight, the following applies:

  — constituentPicture is set equal to 1.

  — If xProjPicture is greater than or equal to pictureWidth, xProjPicture is set to xProjPicture − pictureWidth.

  — If yProjPicture is greater than or equal to pictureHeight, yProjPicture is set to yProjPicture − pictureHeight.

— Otherwise, constituentPicture is set equal to 0.

— Depending on the projection format, the following applies:

  — When the projection format is the equirectangular projection, subclause 5.2.2 is invoked with pictureWidth, pictureHeight, xProjPicture, and yProjPicture as inputs, and the output is assigned to azimuthLocal, elevationLocal.

  — When the projection format is the cubemap projection, subclause 5.2.3 is invoked with pictureWidth, pictureHeight, xProjPicture, and yProjPicture as inputs, and the output is assigned to azimuthLocal, elevantionLocal.

— If RotationFlag is equal to 1, subclause 5.3 is invoked with azimuthLocal, elevantionLocal, `rotation_yaw` $\div 2^{16}$, `rotation_pitch` $\div 2^{16}$, and `rotation_roll` $\div 2^{16}$ as inputs, and the output is assigned to azimuthGlobal and elevationGlobal.

— Otherwise, azimuthGlobal is set equal to azimuthLocal and elevationGlobal is set equal to elevationLocal.

7.5.2 Projection format structure

7.5.2.1 Syntax

```
aligned(8) class ProjectionFormatStruct() {
   bit(3) reserved = 0;
   unsigned int(5) projection_type;
}
```

7.5.2.2 Semantics

projection_type indicates the type of the mapping of the projected picture onto the spherical coordinate system as specified in subclause 5.1. The values of projection_type and their semantics are specified in Table 10.

7.5.3 Region-wise packing structure

7.5.3.1 Definition

RegionWisePackingStruct() specifies the mapping between packed regions and the respective projected regions and specifies the location and size of the guard bands, if any.

NOTE Among other information the RegionWisePackingStruct() also provides the content coverage information in the 2D Cartesian picture domain.

A decoded picture in the semantics of the region-wise packing structure is either one of the following depending on the container for this syntax structure:

— For video, the decoded picture is the decoding output resulting from a sample of the video track.

— For an image item, the decoded picture is a reconstructed image of the image item.

The content of RegionWisePackingStruct() is summarized below, while the semantics follow subsequently in this clause:

— The width and height of the projected picture are explicitly signalled with proj_picture_width and proj_picture_height, respectively.

— The width and height of the packed picture are explicitly signalled with packed_picture_width and packed_picture_height, respectively.

— When the projected picture is stereoscopic and has the top-bottom or side-by-side frame packing arrangement, constituent_picture_matching_flag equal to 1 specifies that

  — the projected region information, packed region information, and guard band region information in this syntax structure apply individually to each constituent picture,

  — the packed picture and the projected picture have the same stereoscopic frame packing format, and

  — the number of projected regions and packed regions is double of that indicated by the value of num_regions in the syntax structure.

— RegionWisePackingStruct() contains a loop, in which a loop entry corresponds to the respective projected regions and packed regions in both constituent pictures (when constituent_picture_matching_flag equal to 1) or to a projected region and the respective packed region (when constituent_picture_matching_flag equal to 0), and the loop entry the contains the following:

  — a flag indicating the presence of guard bands for the packed region,

  — the packing type (however, only rectangular region-wise packing is specified in this document),

  — the mapping between a projected region and the respective packed region in the rectangular region packing structure RectRegionPacking(i),

  — when guard bands are present, the guard band structure for the packed region GuardBand(i).

The content of the rectangular region packing structure RectRegionPacking(i) is summarized below, while the semantics follow subsequently in this clause:

— proj_reg_width[i], proj_reg_height[i], proj_reg_top[i], and proj_reg_left[i] specify the width, height, top offset, and left offset, respectively, of the i-th projected region.

— transform_type[i] specifies the rotation and mirroring, if any, that are applied to the i-th packed region to remap it to the i-th projected region.

— packed_reg_width[i], packed_reg_height[i], packed_reg_top[i], and packed_reg_left[i] specify the width, height, the top offset, and the left offset, respectively, of the i-th packed region.

The content of the guard band structure GuardBand(i)is summarized below, while the semantics follow subsequently in this clause:

— left_gb_width[i], right_gb_width[i], top_gb_height[i], or bottom_gb_height[i] specify the guard band size on the left side of, the right side of, above, or below, respectively, the i-th packed region.

— gb_not_used_for_pred_flag[i] indicates if the encoding was constrained in a manner that guards bands are not used as a reference in the inter prediction process.

— gb_type[i][j] specifies the type of the guard bands for the i-th packed region.

Figure 19 illustrates an example of the position and size of a projected region within a projected picture (on the left side) as well as that of a packed region within a packed picture with guard bands (on the right side). This example applies when the value of constituent_picture_matching_flag is equal to 0.

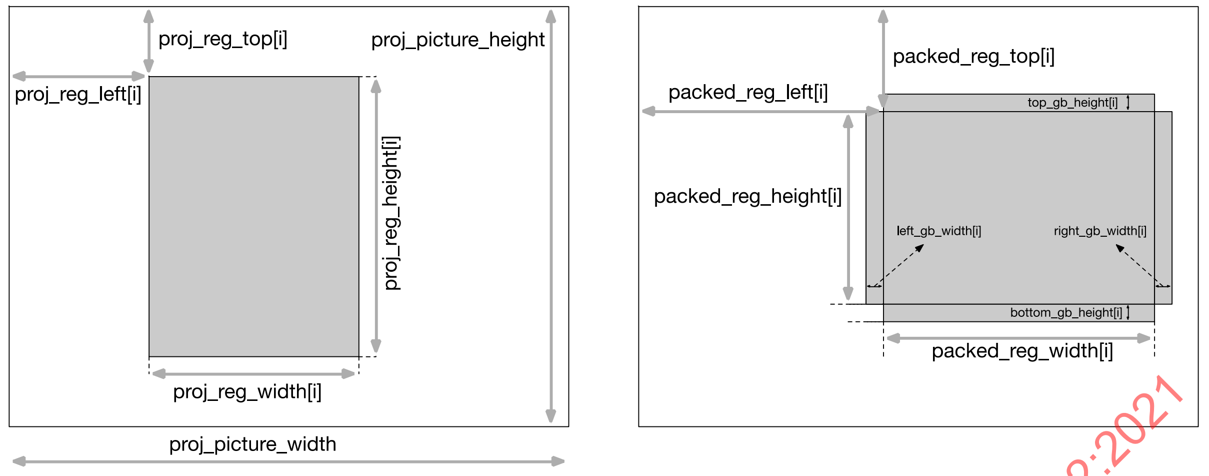


Figure 19 — An example of a projected region and the corresponding packed region with guard bands

The remaining of subclause 7.5.3 is organized as follows:

— The syntax and semantics of the rectangular region packing structure are specified in subclauses 7.5.3.2 and 7.5.3.3, respectively.

— The syntax and semantics of the guard band structure are specified in subclauses 7.5.3.4 and 7.5.3.5, respectively.

— The syntax and semantics of the region-wise packing structure are specified in subclauses 7.5.3.6 and 7.5.3.7, respectively.

— Subclause 7.5.3.8 derives variables from syntax element values of the rectangular region packing, guard band, region-wise packing structures. Subclause 7.5.3.8 also uses the variables to specify constraints for the syntax element values. The variables are also used in other clauses.

7.5.3.2 Syntax of the rectangular region packing structure

```
aligned(8) class RectRegionPacking(i) {
   unsigned int(32) proj_reg_width[i];
   unsigned int(32) proj_reg_height[i];
   unsigned int(32) proj_reg_top[i];
   unsigned int(32) proj_reg_left[i];
   unsigned int(3) transform_type[i];
   bit(5) reserved = 0;
   unsigned int(16) packed_reg_width[i];
   unsigned int(16) packed_reg_height[i];
   unsigned int(16) packed_reg_top[i];
   unsigned int(16) packed_reg_left[i];
}
```

7.5.3.3 Semantics of the rectangular region packing structure

`proj_reg_width[i]`, `proj_reg_height[i]`, `proj_reg_top[i]`, and `proj_reg_left[i]` specify the width, height, top offset, and left offset, respectively, of the i-th projected region, either within the projected picture (when `constituent_picture_matching_flag` is equal to 0) or within the constituent picture of the projected picture (when `constituent_picture_matching_flag` is equal to 1). `proj_reg_width[i]`, `proj_reg_height[i]`, `proj_reg_top[i]` and `proj_reg_left[i]` are indicated in relative projected picture sample units.

NOTE 1 Two projected regions can partially or entirely overlap with each other. When there is an indication of quality difference, e.g. by a region-wise quality ranking indication, then for the overlapping area of any two overlapping projected regions, the packed region corresponding to the projected region that is indicated to have higher quality is suggested to be used for rendering.

`transform_type[i]` specifies the rotation and mirroring that is applied to the i-th packed region to remap it to the i-th projected region. When `transform_type[i]` specifies both rotation and mirroring, rotation is applied before mirroring for converting sample locations of a packed region to sample locations of a projected region. The following values are specified:

0: no transform

1: mirroring horizontally

2: rotation by 180 degrees (counter-clockwise)

3: rotation by 180 degrees (counter-clockwise) before mirroring horizontally

4: rotation by 90 degrees (counter-clockwise) before mirroring horizontally

5: rotation by 90 degrees (counter-clockwise)

6: rotation by 270 degrees (counter-clockwise) before mirroring horizontally

7: rotation by 270 degrees (counter-clockwise)

NOTE 2 Subclause 5.4.2 specifies the semantics of `transform_type[i]` for converting a sample location of a packed region in a packed picture to a sample location of a projected region in a projected picture.

`packed_reg_width[i]`, `packed_reg_height[i]`, `packed_reg_top[i]`, and `packed_reg_left[i]` specify the width, height, the offset, and the left offset, respectively, of the i-th packed region, either within the packed picture (when `constituent_picture_matching_flag` is equal to 0) or within each constituent picture of the packed picture (when `constituent_picture_matching_flag` is equal to 1). `packed_reg_width[i]`, `packed_reg_height[i]`, `packed_reg_top[i]`, and `packed_reg_left[i]` are indicated in relative packed picture sample units. `packed_reg_width[i]`, `packed_reg_height[i]`, `packed_reg_top[i]`, and `packed_reg_left[i]` shall represent integer horizontal and vertical coordinates of luma sample units within the decoded pictures.

NOTE 3 Two packed regions can partially or entirely overlap with each other.

7.5.3.4 Syntax of the guard band structure

```
aligned(8) class GuardBand(i) {
   unsigned int(8) left_gb_width[i];
   unsigned int(8) right_gb_width[i];
   unsigned int(8) top_gb_height[i];
   unsigned int(8) bottom_gb_height[i];
   unsigned int(1) gb_not_used_for_pred_flag[i];
   for (j = 0; j < 4; j++)
      unsigned int(3) gb_type[i][j];
   bit(3) reserved = 0;
}
```

7.5.3.5 Semantics of the guard band structure

`left_gb_width[i]` specifies the width of the guard band on the left side of the i-th packed region in relative packed picture sample units. When the decoded picture has 4:2:0 or 4:2:2 chroma format, `left_gb_width[i]` shall correspond to an even number of luma samples within the decoded picture.

right_gb_width[i] specifies the width of the guard band on the right side of the i-th packed region in relative packed picture sample units. When the decoded picture has 4:2:0 or 4:2:2 chroma format, right_gb_width[i] shall correspond to an even number of luma samples within the decoded picture.

top_gb_height[i] specifies the height of the guard band above the i-th packed region in relative packed picture sample units. When the decoded picture has 4:2:0 chroma format, top_gb_height[i] shall correspond to an even number of luma samples within the decoded picture.

bottom_gb_height[i] specifies the height of the guard band below the i-th packed region in relative packed picture sample units. When the decoded picture has 4:2:0 chroma format, bottom_gb_height[i] shall correspond to an even number of luma samples within the decoded picture.

When GuardBand(i) is present, at least one of left_gb_width[i], right_gb_width[i], top_gb_height[i], or bottom_gb_height[i] shall be greater than 0.

gb_not_used_for_pred_flag[i] equal to 0 specifies that the guard bands may or may not be used in the inter prediction process. gb_not_used_for_pred_flag[i] equal to 1 specifies that the sample values of the guard bands are not used in the inter prediction process.

NOTE 1 When gb_not_used_for_pred_flag[i] is equal to 1, the sample values within guard bands in decoded pictures can be rewritten even if the decoded pictures were used as references for inter prediction of subsequent pictures to be decoded. For example, the content of a packed region can be seamlessly expanded to its guard band with decoded and re-projected samples of another packed region.

gb_type[i][j] specifies the type of the guard bands for the i-th packed region as follows, with j equal to 0, 1, 2, or 3 indicating that the semantics below apply to the left, right, top, or bottom edge, respectively, of the packed region:

— gb_type[i][j] equal to 0 specifies that the content of the guard bands in relation to the content of the packed regions is unspecified. When gb_not_used_for_pred_flag[i] is equal to 0, gb_type[i][j] shall not be equal to 0.

— gb_type[i][j] equal to 1 specifies that the content of the guard bands suffices for interpolation of sub-pixel values within the packed region and less than one pixel outside of the boundary of the packed region.

  NOTE 2 gb_type[i][j] equal to 1 can be used when the boundary samples of a packed region have been copied horizontally or vertically to the guard band.

— gb_type[i][j] equal to 2 specifies that the content of the guard bands represents actual picture content that is spherically adjacent to the content in the packed region and is on the surface of the packed region at quality that gradually changes from the picture quality of the packed region to that of the spherically adjacent packed region.

— gb_type[i][j] equal to 3 specifies that the content of the guard bands represents actual picture content that is spherically adjacent to the content in the packed region and is on the surface of the packed region at the picture quality of the packed region.

— gb_type[i][j] values greater than 3 are reserved.

7.5.3.6 Syntax the region-wise packing structure

```
aligned(8) class RegionWisePackingStruct() {
   unsigned int(1) constituent_picture_matching_flag;
   bit(7) reserved = 0;
   unsigned int(8) num_regions;
   unsigned int(32) proj_picture_width;
   unsigned int(32) proj_picture_height;
   unsigned int(16) packed_picture_width;
   unsigned int(16) packed_picture_height;
   for (i = 0; i < num_regions; i++) {
      bit(3) reserved = 0;
      unsigned int(1) guard_band_flag[i];
      unsigned int(4) packing_type[i];
      if (packing_type[i] == 0) {
         RectRegionPacking(i);
         if (guard_band_flag[i])
            GuardBand(i);
      }
   }
}
```

7.5.3.7 Semantics of the region-wise packing structure

`constituent_picture_matching_flag` equal to 1 specifies that the projected region information, packed region information, and guard band region information in this syntax structure apply individually to each constituent picture and that the packed picture and the projected picture have the same stereoscopic frame packing format. `constituent_picture_matching_flag` equal to 0 specifies that the projected region information, packed region information, and guard band region information in this syntax structure apply to the projected picture. When SpatiallyPackedStereoFlag is equal to 0, `constituent_picture_matching_flag` shall be equal to 0.

NOTE 1 For the stereoscopic content that uses equivalent region-wise packing for the constituent pictures, setting this flag equal to 1 allows more compact signalling of region-wise packing information.

`num_regions` specifies the number of packed regions when `constituent_picture_matching_flag` is equal to 0. Value 0 is reserved. When `constituent_picture_matching_flag` is equal to 1, the total number of packed regions is equal to 2 * `num_regions` and the information in `RectRegionPacking(i)` and `GuardBand(i)` applies to each constituent picture of the projected picture and the packed picture.

`proj_picture_width` and `proj_picture_height` specify the width and height, respectively, of the projected picture, in relative projected picture sample units. `proj_picture_width` and `proj_picture_height` shall both be greater than 0.

NOTE 2 The same sampling grid, width, and height are used for the luma sample array and the chroma sample arrays of the projected picture.

`packed_picture_width` and `packed_picture_height` specify the width and height, respectively, of the packed picture, in relative packed picture sample units. `packed_picture_width` and `packed_picture_height` shall both be greater than 0.

`guard_band_flag[i]` equal to 0 specifies that the i-th packed region has no guard bands. `guard_band_flag[i]` equal to 1 specifies that the i-th packed region has at least one guard band.

`packing_type[i]` specifies the type of region-wise packing. The values of `packing_type[i]` and their semantics are specified in Table 11.

RectRegionPacking(i) specifies the region-wise packing between the i-th packed region and the i-th projected region. The syntax and semantics of RectRegionPacking(i) are specified in subclauses 7.5.3.2 and 7.5.3.3, respectively.

GuardBand(i) specifies the guard bands for the i-th packed region. The syntax and semantics of GuardBand(i) are specified in subclauses 7.5.3.4 and 7.5.3.5, respectively.

7.5.3.8 Derivation of region-wise packing variables and constraints for the syntax elements of the region-wise packing structure

When the i-th packed region as specified by this RegionWisePackingStruct() overlaps with the j-th packed region specified by the same RegionWisePackingStruct(), the i-th and j-th projected regions shall reside in different constituent pictures for any values of i and j that are not equal to each other. The i-th packed region as specified by this RegionWisePackingStruct() shall not overlap with any guard band specified by the same RegionWisePackingStruct().

The guard bands associated with the i-th packed region, if any, as specified by this RegionWisePackingStruct() shall not overlap with any packed region specified by the same RegionWisePackingStruct() or any other guard bands specified by the same RegionWisePackingStruct().

NOTE Projected regions are allowed to overlap. When projected regions overlap and a quality difference is indicated between the projected regions, e.g. by a region-wise quality ranking indication, the packed region that is indicated to have the highest quality among the packed regions corresponding to the projected regions that overlap is expected to be used for rendering the overlapping area.

The variables NumRegions, PackedRegLeft[n], PackedRegTop[n], PackedRegWidth[n], PackedRegHeight[n], ProjRegLeft[n], ProjRegTop[n], ProjRegWidth[n], ProjRegHeight[n], TrasnformType[n], PackingType[n] are derived as follows:

— For n in the range of 0 to num_regions − 1, inclusive, the following applies:

  — PackedRegLeft[n] is set equal to packed_reg_left[n].

  — PackedRegTop[n] is set equal to packed_reg_top[n].

  — PackedRegWidth[n] is set equal to packed_reg_width[n].

  — PackedRegHeight[n] is set equal to packed_reg_height[n].

  — ProjRegLeft[n] is set equal to proj_reg_left[n].

  — ProjRegTop[n] is set equal to proj_reg_top[n].

  — ProjRegWidth[n] is set equal to proj_reg_width[n].

  — ProjRegHeight[n] is set equal to proj_reg_height[n].

  — TransformType[n] is set equal to transform_type[n].

  — PackingType[n] is set equal to packing_type[n].

— If constituent_picture_matching_flag is equal to 0, the following applies:

  — NumRegions is set equal to num_regions.

- Otherwise (constituent_picture_matching_flag is equal to 1), the following applies:
  - NumRegions is set equal to 2 * num_regions.
  - When TopBottomFlag is equal to 1, the following applies:
    - projLeftOffset and packedLeftOffset are both set equal to 0.
    - projTopOffset is set equal to proj_picture_height / 2 and packedTopOffset is set equal to packed_picture_height / 2.
  - When SideBySideFlag is equal to 1, the following applies:
    - projLeftOffset is set equal to proj_picture_width / 2 and packedLeftOffset is set equal to packed_picture_width / 2.
    - projTopOffset and packedTopOffset are both set equal to 0.
  - For n in the range of NumRegions / 2 to NumRegions − 1, inclusive, the following applies:
    - nIdx is set equal to n − NumRegions / 2.
    - PackedRegLeft[n] is set equal to packed_reg_left[nIdx] + packedLeftOffset.
    - PackedRegTop[n] is set equal to packed_reg_top[nIdx] + packedTopOffset.
    - PackedRegWidth[n] is set equal to packed_reg_width[nIdx].
    - PackedRegHeight[n] is set equal to packed_reg_height[nIdx].
    - ProjRegLeft[n] is set equal to proj_reg_left[nIdx] + projLeftOffset.
    - ProjRegTop[n] is set equal to proj_reg_top[nIdx] + projTopOffset.
    - ProjRegWidth[n] is set equal to proj_reg_width[nIdx].
    - ProjRegHeight[n] is set equal to proj_reg_height[nIdx].
    - TransformType[n] is set equal to transform_type[nIdx].
    - PackingType[n] is set equal to packing_type[nIdx].

For each value of n in the range of 0 to NumRegions − 1, inclusive, the values of ProjRegWidth[n], ProjRegHeight[n], ProjRegTop[n], and ProjRegLeft[n] are constrained as follows:

- ProjRegWidth[n] shall be in the range of 1 to proj_picture_width, inclusive.
- ProjRegHeight[n] shall be in the range of 1 to proj_picture_height, inclusive.
- ProjRegLeft[n] shall be in the range of 0 to proj_picture_width − 1, inclusive.
- ProjRegTop[n] shall be in the range of 0 to proj_picture_height − 1, inclusive.
- If ProjRegTop[n] is less than proj_picture_height / VerDiv1, the sum of ProjRegTop[n] and ProjRegHeight[n] shall be less than or equal to proj_picture_height / VerDiv1. Otherwise, the sum of ProjRegTop[n] and ProjRegHeight[n] shall be less than or equal to proj_picture_height / VerDiv1 * 2.

For each value of n in the range of 0 to NumRegions − 1, inclusive, the values of PackedRegWidth[n], PackedRegHeight[n], PackedRegTop[n], and PackedRegLeft[n] are constrained as follows:

— PackedRegWidth[n] shall be in the range of 1 to `packed_picture_width`, inclusive.

— PackedRegHeight[n] shall be in the range of 1 to `packed_picture_height`, inclusive.

— PackedRegLeft[n] shall be in the range of 0 to `packed_picture_width` − 1, inclusive.

— PackedRegTop[n] shall be in the range of 0 to `packed_picture_height` − 1, inclusive.

— If PackedRegLeft[n] is less than `packed_picture_width` / HorDiv1, the sum of PackedRegLeft[n] and PackedRegWidth[n] shall be less than or equal to `packed_picture_width` / HorDiv1. Otherwise, the sum of PackedRegLeft[n] and PackedRegWidth[n] shall be less than or equal to `packed_picture_width` / HorDiv1 * 2.

— If PackedRegTop[n] is less than `packed_picture_height` / VerDiv1, the sum of PackedRegTop[n] and PackedRegHeight[n] shall be less than or equal to `packed_picture_height` / VerDiv1. Otherwise, the sum of PackedRegTop[n] and PackedRegHeight[n] shall be less than or equal to `packed_picture_height` / VerDiv1 * 2.

— When the decoded picture has 4:2:0 or 4:2:2 chroma format, PackedRegLeft[n] shall correspond to an even horizontal coordinate value of luma sample units, and PackedRegWidth[n] shall correspond to an even number of luma samples, both within the decoded picture.

— When the decoded picture has 4:2:0 chroma format, PackedRegTop[n] shall correspond to an even vertical coordinate value of luma sample units, and ProjRegHeight[n] shall correspond to an even number of luma samples, both within the decoded picture.

7.5.4 Rotation structure

7.5.4.1 Definition

The fields in this structure provides the yaw, pitch, and roll angles, respectively, of the rotation to be applied to convert the local coordinate axes to the global coordinate axes. In the case of stereoscopic omnidirectional video, the fields apply to each view individually.

7.5.4.2 Syntax

```
aligned(8) class RotationStruct() {
   signed int(32) rotation_yaw;
   signed int(32) rotation_pitch;
   signed int(32) rotation_roll;
}
```

7.5.4.3 Semantics

`rotation_yaw`, `rotation_pitch`, and `rotation_roll` specify the yaw, pitch, and roll angles, respectively, of the rotation that is applied to the unit sphere to convert the local coordinate axes to the global coordinate axes, in units of 2^{-16} degrees, relative to the global coordinate axes. `rotation_yaw` shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive. `rotation_pitch` shall be in the range of $-90 * 2^{16}$ to $90 * 2^{16}$, inclusive. `rotation_roll` shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive.

7.5.5 Content coverage structure

7.5.5.1 Definition

The fields in this structure provides the content coverage, which is expressed by one or more sphere regions covered by the content, relative to the global coordinate axes.

7.5.5.2 Syntax

```
aligned(8) class ContentCoverageStruct() {
   unsigned int(8) coverage_shape_type;
   unsigned int(8) num_regions;
   unsigned int(1) view_idc_presence_flag;
   if (view_idc_presence_flag == 0) {
      unsigned int(2) default_view_idc;
      bit(5) reserved = 0;
   } else
      bit(7) reserved = 0;
   for ( i = 0; i < num_regions; i++) {
      if (view_idc_presence_flag == 1) {
         unsigned int(2) view_idc[i];
         bit(6) reserved = 0;
      }
      SphereRegionStruct(1, 1);
   }
}
```

7.5.5.3 Semantics

coverage_shape_type specifies the shape of the sphere regions expressing the content coverage. coverage_shape_type has the same semantics as shape_type specified in subclause 7.7.2.3. The value of coverage_shape_type is used as the shape type value when applying subclause 7.5.6 to the semantics of ContentCoverageStruct().

num_regions specifies the number of sphere regions. Value 0 is reserved.

view_idc_presence_flag equal to 0 specifies that view_idc[i] is not present. view_idc_presence_flag equal to 1 specifies that view_idc[i] is present and indicates the association of sphere regions with particular (left, right, or both) views.

default_view_idc equal to 0 indicates that each sphere region is monoscopic, 1 indicates that each sphere region is on the left view of a stereoscopic content, 2 indicates that each sphere region is on the right view of a stereoscopic content, 3 indicates that each sphere region is on both the left and right views.

view_idc[i] equal to 1 indicates that the i-th sphere region is on the left view of a stereoscopic content, 2 indicates the i-th sphere region is on the right view of a stereoscopic content, and 3 indicates that the i-th sphere region is on both the left and right views. view_idc[i] equal to 0 is reserved.

NOTE view_idc_presence_flag equal to 1 enables indicating asymmetric stereoscopic coverage. For example, one example of an asymmetric stereoscopic coverage can be described by setting num_regions equal to 2, indicating one sphere region to be on the left view covering the azimuth range of −90° to 90°, inclusive, and indicating the other sphere region to be on the right view covering the azimuth range of −60 to 60°, inclusive.

When SphereRegionStruct(1, 1) is included in the ContentCoverageStruct(), subclause 7.5.6 applies and interpolate shall be equal to 0.

The content coverage is specified by the union of num_regions SphereRegionStruct(1, 1) structure(s). When num_regions is greater than 1, the content coverage may be non-contiguous.

7.5.6 Sphere region structure

7.5.6.1 Definition

The sphere region structure (`SphereRegionStruct`) specifies a sphere region.

When `centre_tilt` is equal to 0, the sphere region specified by this structure is derived as follows:

— If both `azimuth_range` and `elevation_range` are equal to 0, the sphere region specified by this structure is a point on a spherical surface.

— Otherwise, the variables centreAzimuth and centreElevation are derived as follows:

$$\text{centreAzimuth} = \texttt{centre\_azimuth} \div 65536 \qquad (15)$$
$$\text{centreElevation} = \texttt{centre\_elevation} \div 65536$$

The sphere region is defined as follows with reference to the shape type value specified in the semantics of the structure containing this instance of `SphereRegionStruct`:

— When the shape type value is equal to 0, the sphere region is specified by four great circles defined by four points cAzimuth1, cAzimuth2, cElevation1, cElevation2 and the centre point defined by centreAzimuth equal to 0 and centreElevation equal to 0 as shown in Figure 20a and the variable derivation is as follows:

$$\text{cAzimuth1} = (\ 0 - \texttt{azimuth\_range} \div 2\ ) \div 65536 \qquad (16)$$
$$\text{cAzimuth2} = (\ 0 + \texttt{azimuth\_range} \div 2\ ) \div 65536$$
$$\text{cElevation1} = (\ 0 - \texttt{elevation\_range} \div 2\ ) \div 65536$$
$$\text{cElevation2} = (\ 0 + \texttt{elevation\_range} \div 2\ ) \div 65536$$

Subsequently, the sphere region is rotated by centreAzimuth degrees in the azimuth direction, and by centreElevation degrees in the elevation direction, while its shape remains the same.

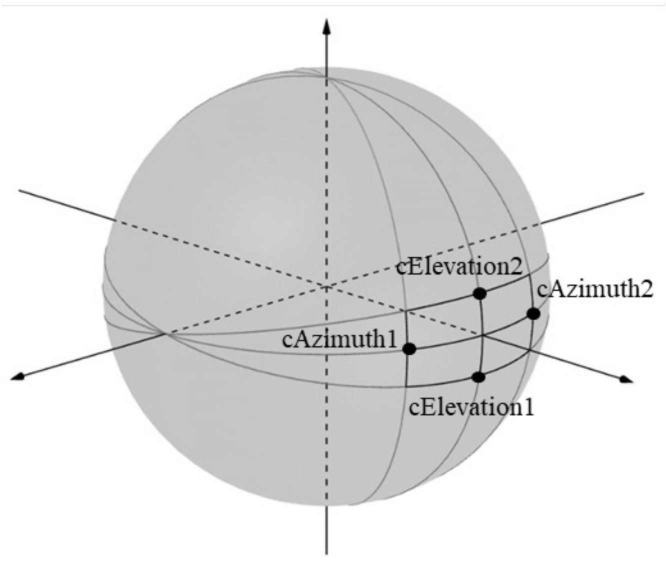
— When the shape type value is equal to 1, the sphere region is specified by two azimuth circles and two elevation circles defined by four points cAzimuth1, cAzimuth2, cElevation1, cElevation2 and the centre point defined by centreAzimuth and centreElevation and as shown in Figure 21 and the variable derivation is as follows:

$$\text{cAzimuth1} = (\ \texttt{centre\_azimuth} - \texttt{azimuth\_range} \div 2\ ) \div 65536 \qquad (17)$$
$$\text{cAzimuth2} = (\ \texttt{centre\_azimuth} + \texttt{azimuth\_range} \div 2\ ) \div 65536$$
$$\text{cElevation1} = (\ \texttt{centre\_elevation} - \texttt{elevation\_range} \div 2\ ) \div 65536$$
$$\text{cElevation2} = (\ \texttt{centre\_elevation} + \texttt{elevation\_range} \div 2\ ) \div 65536$$

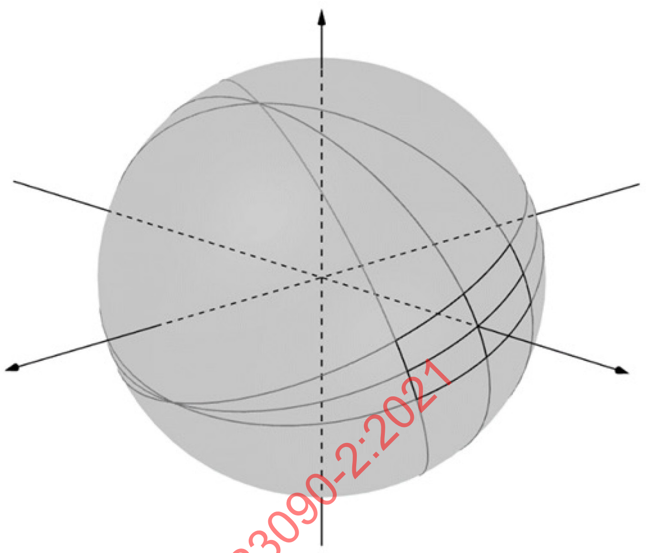
When `centre_tilt` is not equal to 0, the sphere region is firstly derived as above and then a tilt rotation is applied along the axis originating from the sphere origin passing through the centre point of the sphere region, where the angle value increases clockwise when looking from the origin towards the positive end of the axis. The final sphere region is the one after applying the tilt rotation.

Shape type value equal to 0 specifies that the sphere region is specified by four great circles.

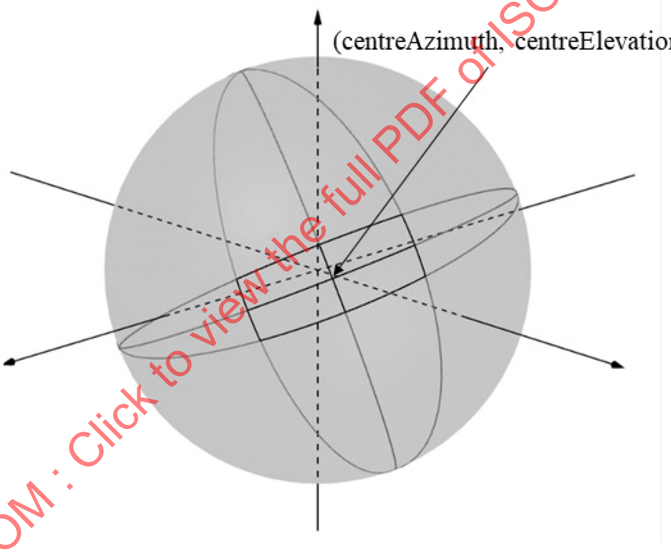
NOTE When shape type value is equal to 0, the order of applying the rotation by centreAzimuth and centreElevation and the tilting by `centre_tilt` can be switched without affecting the resulting sphere region. Figure 20 illustrates a process for deriving of a sphere region with shape type equal to 0 by first tilting the sphere region by `centre_tilt` and then applying centreAzimuth and centreElevation to the tilted sphere region.



a) sphere region with centre point azimuth and elevation equal to 0



b) Tilting by `centre_tilt`



c) rotating the sphere region by centreAzimuth and centreElevation

Figure 20 — Deriving a sphere region specified by four great circles

Shape type value equal to 1 specifies that the sphere region is specified by two azimuth circles and two elevation circles as illustrated in Figure 21.

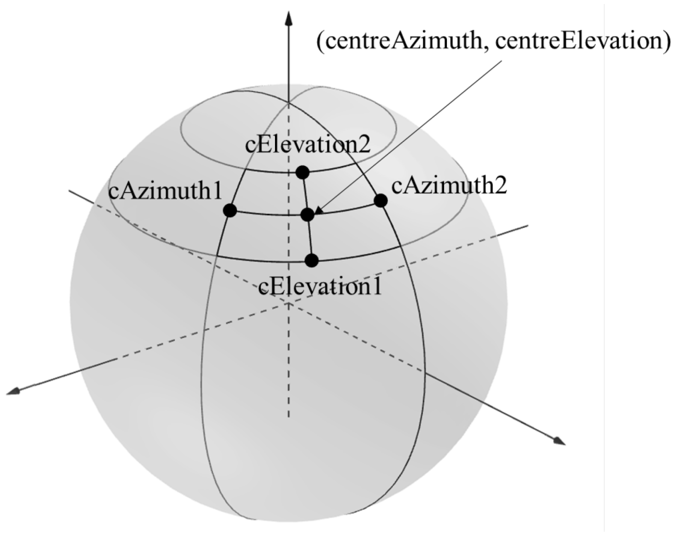


Figure 21 — A sphere region specified by two azimuth circles and two elevation circles

Shape type values greater than 1 are reserved.

7.5.6.2 Syntax

```
aligned(8) SphereRegionStruct(range_included_flag, interpolate_included_flag) {
   signed int(32) centre_azimuth;
   signed int(32) centre_elevation;
   signed int(32) centre_tilt;
   if (range_included_flag) {
      unsigned int(32) azimuth_range;
      unsigned int(32) elevation_range;
   }
   if (interpolate_included_flag) {
      unsigned int(1) interpolate;
      bit(7) reserved = 0;
   }
}
```

7.5.6.3 Semantics

`centre_azimuth` and `centre_elevation` specify the azimuth and elevation values, respectively, of the centre of the sphere region in units of 2^{-16} degrees. `centre_azimuth` shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive. `centre_elevation` shall be in the range of $-90 * 2^{16}$ to $90 * 2^{16}$, inclusive.

`centre_tilt` specifies the tilt angle of the sphere region in units of 2^{-16} degrees. `centre_tilt` shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive.

`azimuth_range` and `elevation_range`, when present, specify the azimuth and elevation ranges, respectively, of the sphere region specified by this structure in units of 2^{-16} degrees. `azimuth_range` and `elevation_range` specify the range through the centre point of the sphere region, as illustrated by Figure 20 or Figure 21. When `azimuth_range` and `elevation_range` are not present in this instance of `SphereRegionStruct()`, they are inferred as specified in the semantics of the structure containing this instance of `SphereRegionStruct()`. When the shape type value is equal to 0, `azimuth_range` shall be in the range of 0 to $180 * 2^{16}$, inclusive. When the shape type value is equal to 1, `azimuth_range` shall be in the range of 0 to $360 * 2^{16}$, inclusive. `elevation_range` shall be in the range of 0 to $180 * 2^{16}$, inclusive.

NOTE When the shape type value is equal to 0, azimuth_range is required to be in the range of 0 to 180 * 2^{16}, inclusive, since otherwise the shape no longer corresponds to a rectangle that is presented straight in front of the viewer.

The semantics of interpolate are specified by the semantics of the structure containing this instance of SphereRegionStruct(). When interpolate is not present in this instance of SphereRegionStruct(), it is inferred as specified in the semantics of the syntax structure containing this instance of SphereRegionStruct().

7.6 Restricted video schemes for omnidirectional video

7.6.1 Scheme types

7.6.1.1 Open-ended and closed scheme types

An open-ended scheme type for a kind of transformation is a scheme type that allows future extensions. For example, the 'stvi' scheme type (stereoscopic video) may be used for a new frame packing arrangement type that is defined after the definition of the 'stvi' scheme type. A closed scheme type, on the other hand, when defined, clearly specifies which transformations are allowed and does not allow future extensions.

SchemeTypeBox allows inclusion of only one scheme type and there may be only one instance of SchemeTypeBox included in RestrictedSchemeInfoBox. The scheme type included in SchemeTypeBox shall be an open-ended scheme type, i.e. 'podv' or 'fodv' or 'modv'.

CompatibleSchemeTypeBox allows inclusion of only one scheme type but there may be multiple instances of CompatibleSchemeTypeBox included in RestrictedSchemeInfoBox. A closed scheme type shall only be included in an instance of CompatibleSchemeTypeBox.

An OMAF player that does not recognize the open-ended scheme type in SchemeTypeBox shall ignore the track. An OMAF player that recognizes the open-ended scheme type in SchemeTypeBox but none of the scheme types in the instances of CompatibleSchemeTypeBox, if any, should parse all boxes contained in SchemeInformationBox to determine whether it has the capability required to properly process the track. An OMAF player should ignore the track unless it supports all boxes contained in SchemeInformationBox and all syntax elements and syntax element values present in those boxes.

7.6.1.2 Projected omnidirectional video ('podv')

The use of the projected omnidirectional video scheme for the restricted video sample entry type 'resv' indicates that the decoded pictures are packed pictures containing either monoscopic or stereoscopic content. The use of the projected omnidirectional video scheme is indicated by scheme_type equal to 'podv' (projected omnidirectional video) within SchemeTypeBox in the RestrictedSchemeInfoBox.

The format of the projected monoscopic pictures is indicated with the ProjectedOmniVideoBox contained within the SchemeInformationBox. One and only one ProjectedOmniVideoBox shall be present in the SchemeInformationBox when the scheme type is 'podv'.

The 'podv' scheme type is defined as an open-ended scheme type for projected omnidirectional video.

As specified in subclause 7.6.2, a ProjectionFormatBox shall be present within the ProjectedOmniVideoBox. ProjectionFormatBox is not constrained beyond the specification in subclause 7.6.2. The 'podv' scheme type may be used with all version values specified for ProjectionFormatBox in this document and in any of its future amendments and editions. The 'podv' scheme type may be used with any projection_type value specified in this document and in any of its future amendments and editions.

When the ProjectedOmniVideoBox is present in the SchemeInformationBox, StereoVideoBox may be present in the same SchemeInformationBox.

For stereoscopic video, the frame packing arrangement of the projected left and right pictures is indicated with the StereoVideoBox contained within the SchemeInformationBox. The absence of StereoVideoBox indicates that the omnidirectionally projected content of the track is monoscopic. When StereoVideoBox is present in the SchemeInformationBox for the omnidirectional video scheme, version shall be equal to 0, stereo_scheme shall be equal to 4 and the first byte of stereo_indication_type shall be equal to 3, 4, or 5 indicating that the side-by-side frame packing, the top-bottom frame packing, or the temporal interleaving of alternating first and second constituent frames, respectively, is in use and the second byte of stereo_indication_type shall be equal to 0 indicating that quincunx sampling is not in use.

NOTE The 'stvi' scheme type is not expected to be used when the 'podv' scheme type is used.

Optional region-wise packing is indicated with the RegionWisePackingBox contained within the ProjectedOmniVideoBox. The absence of RegionWisePackingBox indicates that no region-wise packing is applied, i.e. that the packed picture is identical to the projected picture.

RegionWisePackingBox is not constrained beyond the specification in subclause 7.6.4. The 'podv' scheme type may be used with all version values specified for RegionWisePackingBox in this document and in any of its future amendments and editions. The 'podv' scheme type may be used with any values of the syntax elements of RegionWisePackingBox specified and allowed in this document and in any of its future amendments and editions.

In addition to the boxes constrained above, SchemeInformationBox may directly or indirectly contain any boxes allowed by this document and in any of its future amendments and editions. Those boxes are not constrained beyond their definition, syntax, and semantics.

7.6.1.3 Equirectangular projected video ('erpv')

NOTE This scheme type can be used for specifying media profiles.

The 'erpv' scheme type is defined as a closed scheme type for projected omnidirectional video.

When scheme_type is equal to 'erpv' in an instance of CompatibleSchemeTypeBox in the RestrictedSchemeInfoBox, the track conforms to the constraints of scheme_type equal to 'podv' with all of the following additional constraints:

- ProjectionFormatBox within the ProjectedOmniVideoBox shall indicate the equirectangular projection.
- When RegionWisePackingBox is present, the following constraints all apply:
  - The value of NumRegions shall be equal to HorDiv1 * VerDiv1.
  - For each value of i in the range of 0 to NumRegions − 1, inclusive, the following applies:
    - The value of PackingType[i] shall be equal to 0.
    - The value of TransformType[i] shall be equal to 0.
    - The value of PackedRegWidth[i] shall be equal to ProjRegWidth[i].
    - The value of PackedRegHeight[i] shall be equal to ProjRegHeight[i].

— version of ProjectionFormatBox, StereoVideoBox (when present), RegionWisePackingBox (when present), RotationBox (when present), and CoverageInformationBox (when present) shall be equal to 0.

— SchemeInformationBox shall not directly or indirectly contain any boxes other than ProjectedOmniVideoBox, ProjectionFormatBox, StereoVideoBox, RegionWisePackingBox, RotationBox, and CoverageInformationBox.

7.6.1.4 Packed equirectangular or cubemap projected video ('ercm')

NOTE This scheme type can be used for specifying media profiles.

The 'ercm' scheme type is defined as a closed scheme type for projected omnidirectional video.

When scheme_type is equal to 'ercm' in an instance of CompatibleSchemeTypeBox in the RestrictedSchemeInfoBox, the track conforms to the constraints of scheme_type equal to 'podv', scheme_type equal to 'podv' shall be present in SchemeTypeBox in the RestrictedSchemeInfoBox, and all of the following additional constraints apply:

— ProjectionFormatBox within the ProjectedOmniVideoBox shall indicate either the equirectangular projection or the cubemap projection.

— When RegionWisePackingBox is present, the value of packing_type[i] for each value of i shall be equal to 0.

— version of ProjectionFormatBox, StereoVideoBox (when present), RegionWisePackingBox (when present), RotationBox (when present), and CoverageInformationBox (when present) shall be equal to 0.

— SchemeInformationBox shall not directly or indirectly contain any boxes other than ProjectedOmniVideoBox, ProjectionFormatBox, StereoVideoBox, RegionWisePackingBox, RotationBox, and CoverageInformationBox.

7.6.1.5 Fisheye omnidirectional video ('fodv')

The use of the fisheye omnidirectional video scheme for the restricted video sample entry type 'resv' indicates that the decoded pictures are fisheye video pictures. The use of the fisheye omnidirectional video scheme is indicated by scheme_type equal to 'fodv' (fisheye omnidirectional video) within SchemeTypeBox in the RestrictedSchemeInfoBox.

The format of fisheye video is indicated with the FisheyeOmniVideoBox contained within the SchemeInformationBox. One and only one FisheyeOmniVideoBox shall be present in the SchemeInformationBox when the scheme type is 'fodv' within SchemeTypeBox in the RestrictedSchemeInfoBox.

The 'fodv' scheme type is defined as an open-ended scheme type for fisheye omnidirectional video.

The 'fodv' scheme type may be used with all version values specified for FisheyeVideoEssentialInfoBox and FisheyeVideoSupplementalInfoBox in this document and in any of its future amendments and editions. The 'fodv' scheme type may be used with any values of syntax elements of FisheyeOmniVideoBox allowed in this document and in any of its future amendments and editions.

When FisheyeOmniVideoBox is present in the SchemeInformationBox, StereoVideoBox shall not be present in the same SchemeInformationBox.

In addition to the boxes constrained above, SchemeInformationBox may directly or indirectly contain any boxes allowed by this document and in any of its future amendments and editions. Those boxes are not constrained beyond their definition, syntax, and semantics.

7.6.1.6 Equirectangular or cubemap projected video with overlays ('ecov')

When scheme_type is equal to 'ecov' in an instance of CompatibleSchemeTypeBox in the RestrictedSchemeInfoBox, the track conforms to the constraints of scheme_type equal to 'ercm' except that the ProjectedOmniVideoBox contained in the SchemeInformationBox is additionally allowed to contain OverlayConfigBox. The value of version of OverlayConfigBox (when present) shall be equal to 0.

7.6.1.7 Packed equirectangular or cubemap projected video with derived region-wise packing ('erc2')

NOTE 1 This scheme type can be used for specifying media profiles.

The 'erc2' scheme type is defined as a closed scheme type for projected omnidirectional video.

When scheme_type is equal to 'erc2' in an instance of CompatibleSchemeTypeBox in the RestrictedSchemeInfoBox, the track conforms to the constraints of scheme_type equal to 'podv', scheme_type equal to 'podv' shall be present in SchemeTypeBox in the RestrictedSchemeInfoBox, and all of the following additional constraints apply:

— ProjectionFormatBox within the ProjectedOmniVideoBox shall indicate either the equirectangular projection or the cubemap projection.

— RegionWisePackingBox shall be present in each media track referenced by the track containing the 'erc2' scheme type. The following constraints shall be obeyed in each RegionWisePackingBox:

  — The value of packing_type[i] for each value of i shall be equal to 0.

  — version shall be equal to 0.

  — When the cubemap projection is in use, proj_reg_top[i], proj_reg_left[i], proj_reg_width[i], and proj_reg_height[i] shall be such that the projected region is within one cube face for all values of i in the range of 0 to num_regions − 1, inclusive.

  — When the equirectangular projection is in use and the content is stereoscopic, proj_reg_top[i], proj_reg_left[i], proj_reg_width[i], and proj_reg_height[i] shall be such that the projected region is within one constituent picture, i.e. not spanning to both left and right views, for all values of i in the range of 0 to num_regions − 1, inclusive.

  — The i-th projected region shall not overlap with the j-th projected region for any unequal values of i and j in the range of 0 to num_regions − 1, inclusive.

— RegionWisePackingBox shall be empty in the track containing the 'erc2' scheme type. Players shall derive the region-wise packing format of the track containing the 'erc2' scheme type from the collection of the instances of RegionWisePackingBox of the tracks referenced by the track containing the 'erc2' scheme type. When no RegionWisePackingBox is present in a track referenced by the track containing the 'erc2' scheme type, players shall conclude that the respective region does not contain any packed regions.

  NOTE 2 For example, an overlay can be carried in a referenced track with no RegionWisePackingBox.

— version of ProjectionFormatBox, StereoVideoBox (when present), RotationBox (when present), and CoverageInformationBox (when present) shall be equal to 0.

— SchemeInformationBox in the track containing the 'erc2' scheme type shall not directly or indirectly contain any boxes other than ProjectedOmniVideoBox, ProjectionFormatBox, StereoVideoBox, RotationBox, and CoverageInformationBox.

7.6.1.8 Open-ended scheme type for mesh omnidirectional video ('modv')

The use of the mesh omnidirectional video scheme for the restricted video sample entry type 'resv' indicates that the decoded pictures consist of rectangular regions that map to mesh elements of a 3D mesh. The use of the mesh omnidirectional video scheme is indicated by scheme_type equal to 'modv' (mesh omnidirectional video) within SchemeTypeBox in the RestrictedSchemeInfoBox.

The 'modv' scheme type is defined as an open-ended scheme type for mesh omnidirectional video.

At least one of the following shall be true:

— One and only one MeshOmniVideoBox is present in the SchemeInformationBox. The 3D mesh is indicated with the MeshBox contained in the MeshOmniVideoBox within the SchemeInformationBox.

— The track contains a tile mesh sample group as specified in subclause 7.16.

When a MeshOmniVideoBox is present within the SchemeInformationBox, a MeshBox shall be present within the MeshOmniVideoBox. The MeshBox is not constrained beyond the specification in subclause 7.6.8. The 'modv' scheme type may be used with all version values specified for MeshBox in this document and in any of its future amendments and editions. The 'modv' scheme type may be used with any mesh_type value specified in this document and in any of its future amendments and editions.

When MeshOmniVideoBox is present in the SchemeInformationBox, StereoVideoBox may be present in the same SchemeInformationBox. The absence of StereoVideoBox indicates that the omnidirectionally projected content of the track is monoscopic. The presence of StereoVideoBox indicates that the omnidirectionally projected content of the track is stereoscopic. When StereoVideoBox is present in the SchemeInformationBox for the mesh omnidirectional video scheme, version shall be equal to 0. The frame packing format indicated in the StereoVideoBox should be ignored as it may or may not correspond to how the stereoscopic content is arranged in the decoded pictures. The mapping of rectangular regions for the left and right views is specified with the tile mesh sample group.

In addition to the boxes constrained above, SchemeInformationBox may directly or indirectly contain any boxes allowed by this document and in any of its future amendments and editions. Those boxes are not constrained beyond their definition, syntax, and semantics.

NOTE It is possible to author tracks that comply with both the 'modv' and 'podv' scheme types. For example, a complete or partial cubemap can be represented by both 'modv' and 'podv' scheme types. When a track complies with both the 'modv' and 'podv' scheme types, one of 'modv' and 'podv' can be present in the SchemeTypeBox, while the other can be present in the CompatibleSchemeTypeBox. An OMAF player can perform processing for either mesh or projected omnidirectional video to render the content.

7.6.1.9 Closed scheme type for mesh omnidirectional video ('meov')

NOTE This scheme type can be used for specifying media profiles.

The 'meov' scheme type is defined as a closed scheme type for mesh omnidirectional video.

When scheme_type is equal to 'meov' in an instance of CompatibleSchemeTypeBox in the RestrictedSchemeInfoBox, the track conforms to the constraints of scheme_type equal to 'modv', scheme_type equal to 'modv' shall be present in SchemeTypeBox or CompatibleSchemeTypeBox in the RestrictedSchemeInfoBox, and all of the following additional constraints apply:

— version of MeshBox and RotationBox (when present) shall be equal to 0.

— SchemeInformationBox shall not directly contain any boxes other than MeshOmniVideoBox, and StereoVideoBox.

— MeshOmniVideoBox shall not directly contain any boxes other than MeshBox and RotationBox.

7.6.2 Projected omnidirectional video box

7.6.2.1 Definition

Box Type:	'povd'
Container:	SchemeInformationBox
Mandatory:	Yes, when scheme_type is equal to 'podv'
Quantity:	Zero or one

This box is a container box that contains boxes indicating information for the following:

— the projection format of the projected picture (C for monoscopic video contained in the track, C_L and C_R for left and right view of stereoscopic video),

— region-wise packing, when applicable,

— the rotation for conversion between the local coordinate axes and the global coordinate axes, if applied, and

— optionally the content coverage of the track.

The values of the variables HorDiv1 and VerDiv1 are set as follows:

— If StereoVideoBox is not present in SchemeInformationBox, HorDiv1 is set equal to 1 and VerDiv1 is set equal to 1.

— Otherwise (StereoVideoBox is present in SchemeInformationBox), the following applies:

  — If side-by-side frame packing is indicated, HorDiv1 is set equal to 2 and VerDiv1 is set equal to 1.

  — Otherwise if top-bottom frame packing is indicated, HorDiv1 is set equal to 1 and VerDiv1 is set equal to 2.

  — Otherwise (temporal interleaving is indicated), HorDiv1 and VerDiv1 are both set equal to 1.

If RotationBox is not present in ProjectedOmniVideoBox, RotationFlag is set equal to 0. Otherwise, RotationFlag is set equal to 1.

If StereoVideoBox is not present in SchemeInformationBox, SpatiallyPackedStereoFlag, TopBottomFlag, and SideBySideFlag are set equal to 0. Otherwise, the following applies:

— When the StereoVideoBox indicates top-bottom frame packing, SpatiallyPackedStereoFlag is set equal to 1, TopBottomFlag is set equal to 1, and SideBySideFlag is set equal to 0.

— When the StereoVideoBox indicates side-by-side frame packing, SpatiallyPackedStereoFlag is set equal to 1, TopBottomFlag is set equal to 0, and SideBySideFlag is set equal to 1.

— When the StereoVideoBox indicates temporal interleaving, SpatiallyPackedStereoFlag, TopBottomFlag, and SideBySideFlag are all set equal to 0.

The following applies:

— The width and height of a monoscopic projected luma picture (ConstituentPicWidth and ConstituentPicHeight, respectively) are derived as follows:

  — If RegionWisePackingBox is not present in ProjectedOmniVideoBox, ConstituentPicWidth and ConstituentPicHeight are set to be equal to width / HorDiv1 and height / VerDiv1, respectively, where width and height are syntax elements of VisualSampleEntry.

  — Otherwise, ConstituentPicWidth and ConstituentPicHeight are set equal to proj_picture_width / HorDiv1 and proj_picture_height / VerDiv1, respectively.

— If RegionWisePackingBox is not present in ProjectedOmniVideoBox, RegionWisePackingFlag is set equal to 0. Otherwise, RegionWisePackingFlag is set equal to 1.

— The semantics of the sample locations of each decoded picture resulting by decoding the samples referring to this sample entry are specified in subclause 7.5.1.2.

7.6.2.2 Syntax

```
aligned(8) class ProjectedOmniVideoBox extends Box('povd') {
   ProjectionFormatBox(); // mandatory
   // optional boxes but no fields
}

aligned(8) class ProjectionFormatBox() extends FullBox('prfr', 0, 0) {
   ProjectionFormatStruct();
}
```

7.6.3 Fisheye omnidirectional video box

7.6.3.1 Definition

Box Type:	'fovd'
Container:	SchemeInformationBox
Mandatory:	Yes, when scheme_type is equal to 'fodv'
Quantity:	Zero or one

FisheyeOmniVideoBox provides essential fisheye video parameters for stitching and rendering of fisheye video at the OMAF player. The fields in FisheyeOmniVideoBox provide region information of circular images in the coded picture and field of view and camera parameters of fisheye lens. FisheyeVideoSupplementalInfoBox provides the local field of view information for high quality stitching and rendering of fisheye video, as well as deadzone information.

This document specifies FisheyeVideoSupplementalInfoBox with version equal to 0 or 1.

7.6.3.2 Syntax

```
aligned(8) class FisheyeOmniVideoBox extends Box('fovd') {
   FisheyeVideoEssentialInfoBox(); // mandatory
   FisheyeVideoSupplementalInfoBox(); // optional
}

aligned(8) class FisheyeVideoEssentialInfoBox extends FullBox('fovi', 0, 0) {
   FisheyeVideoEssentialInfoStruct();
}
```

```
aligned(8) class FisheyeVideoSupplementalInfoBox
                    extends FullBox('fvsi', version, 0) {
   FisheyeVideoSupplementalInfoStruct(version);
}
```

7.6.4 Region-wise packing box

7.6.4.1 Definition

Box Type:	`'rwpk'`
Container:	`ProjectedOmniVideoBox`
Mandatory:	No
Quantity:	Zero or one

`RegionWisePackingBox` specifies the mapping between packed regions and the corresponding projected regions and specifies the location and size of the guard bands, if any.

`RegionWisePackingBox` may be empty, i.e. contain only the box header and no payload. An empty `RegionWisePackingBox` indicates that the region-wise packing format of this track is derived from the collection of the instances of `RegionWisePackingBox` of the tracks referenced by this track.

NOTE Among other information a non-empty `RegionWisePackingBox` also provides the content coverage information in the 2D Cartesian picture domain.

7.6.4.2 Syntax

```
aligned(8) class RegionWisePackingBox extends FullBox('rwpk', 0, 0) {
   if ((size == 1  && largesize > 16) || (size > 12))
      RegionWisePackingStruct();
}
```

7.6.4.3 Semantics

Subclause 7.5.3 applies with the following additional constraint:

— `packed_picture_width` and `packed_picture_height` shall have such values that `packed_picture_width` is an integer multiple of `width` and `packed_picture_height` is an integer multiple of `height`, where `width` and `height` are syntax elements of the `VisualSampleEntry` containing this box.

7.6.5 Rotation box

7.6.5.1 Definition

Box Type:	`'rotn'`
Container:	`ProjectedOmniVideoBox` or `MeshOmniVideoBox`
Mandatory:	No
Quantity:	Zero or one

The fields in this box provides the yaw, pitch, and roll angles, respectively, of the rotation to be applied to convert the local coordinate axes to the global coordinate axes. In the case of stereoscopic omnidirectional video, the fields apply to each view individually. When the `RotationBox` is not present, the fields `rotation_yaw`, `rotation_pitch`, and `rotation_roll` are all inferred to be equal to 0.

7.6.5.2 Syntax

```
aligned(8) class RotationBox extends FullBox('rotn', 0, 0) {
   RotationStruct();
}
```

7.6.6 Coverage information box

7.6.6.1 Definition

Box Type:	`'covi'`
Container:	`ProjectedOmniVideoBox` or `SpatialRelationship2DDescriptionBox`
Mandatory:	No
Quantity:	Zero or one

This box provides information on the content coverage of this track (when the box is contained in `ProjectedOmniVideoBox`) or the content coverage of the composition pictures (when the box is contained in `SpatialRelationship2DDescriptionBox`).

NOTE 1 It is totally up to the OMAF player to handle the area that is not covered by the content when rendering the omnidirectional video content.

NOTE 2 When this box is contained in `ProjectedOmniVideoBox`, this box can be used to indicate whether the sub-picture carried in the track is monoscopic or stereoscopic by the syntax elements `default_view_idc` or `view_idc[i]`.

Each sphere location within the sphere regions specifying the content coverage shall have a corresponding sample in the decoded pictures (when the box is contained in `ProjectedOmniVideoBox`) or in the composition pictures (when the box is contained in `SpatialRelationship2DDescriptionBox`). However, there may be some sphere locations that do have corresponding samples in the decoded pictures but are outside the content coverage.

7.6.6.2 Syntax

```
aligned(8) class CoverageInformationBox extends FullBox('covi', 0, 0) {
   ContentCoverageStruct()
}
```

7.6.7 Mesh omnidirectional video box

7.6.7.1 Definition

Box Type:	`'movd'`
Container:	`SchemeInformationBox`
Mandatory:	Yes, when `scheme_type` is equal to `'modv'`
Quantity:	Zero or one

This box is a container box that contains boxes indicating information for the following:

— the 3D mesh, consisting of mesh elements, and

— the rotation for conversion between the local coordinate axes and the global coordinate axes, if applied.

7.6.7.2 Syntax

```
aligned(8) class MeshOmniVideoBox extends Box('movd') {
   // boxes but no fields
}
```

7.6.8 Mesh box

7.6.8.1 Definition

Box Type:	`'mesh'`
Container:	`MeshOmniVideoBox`
Mandatory:	Yes
Quantity:	One

`MeshBox` specifies a 3D mesh consisting of one or more mesh elements, each of which have a unique ID. The different mesh elements are referenced by one or more tile mesh group entries (e.g. instances of `TileMeshGroupEntry`), which define the relationship between a group of samples (or a complete OMAF tile track) and the position on the mesh. This information can be used for both selection of OMAF tile tracks as well as during the rendering process.

The 3D mesh specified by a `MeshBox` is indicated relative to the local coordinate axes.

NOTE 1 `RotationBox`, when present, specifies the conversion from the local coordinate axes to the global coordinate axes.

In line with how typical rendering engines work with stereoscopic content, the same mesh is assumed to be used for both eyes in case of stereoscopic content. In other words, it is not possible to define distinct left-eye and right-eye meshes.

When a `MeshBox` is present in an OMAF base track, each mesh element in the `MeshBox` shall be mapped into a rectangular region of a decoded picture within one or more referenced OMAF tile tracks using the tile mesh sample grouping.

NOTE 2 Among other information, the `MeshBox` also provides implicit content coverage information.

7.6.8.2 Syntax

```
aligned(8) class MeshBox() extends FullBox ('mesh', 0, 0) {
   unsigned int(16) num_mesh_elements;
   unsigned int(4) mesh_type;
   unsigned int(1) explicit_mesh_id_flag;
   unsigned int(1) explicit_mesh_shape_flag;
   unsigned int(2) reserved;
   if (explicit_mesh_shape_flag)
      unsigned int (8) mesh_shape;
   for (i=0; i < num_mesh_elements; i++) {
      if (mesh_type == 0 || mesh_type == 2)
         SphereRegionStruct(1,0);
      else if (mesh_type == 1)
         3DParallelogramStruct();
      if (explicit_mesh_id_flag == 1)
         unsigned int(16) mesh_element_id;
   }
}
```

7.6.8.3 Semantics

`num_mesh_elements` specifies the number of mesh elements described in this box.

`mesh_type` specifies the type of the mesh description used all mesh elements described in this box. `mesh_type` shall be in the range of to 0 to 2, inclusive. Other values of `mesh_type` are reserved.

`explicit_mesh_id_flag` equal to 0 specifies that `mesh_element_id` is not present in the `MeshBox` and the values of `mesh_element_id` are inferred. `explicit_mesh_id_flag` equal to 1 specifies that `mesh_element_id` is present in the `MeshBox`.

`explicit_mesh_shape_flag` equal to 1 specifies that `mesh_shape` is present in the `MeshBox`. `explicit_mesh_shape_flag` equal to 0 specifies that the `mesh_shape` is not present in the `MeshBox`, in that case no global shape can be presupposed for the mesh described by the `MeshBox`.

`mesh_shape` equal to 0 indicates the `MeshBox` describes a sphere or a portion of a sphere. `mesh_shape` equal to 1 indicates that the `MeshBox` describes a cube or a portion of a cube. `mesh_shape` equal to 2 indicates the `MeshBox` describes a portion of a plane. Other values of `mesh_shape` are reserved.

`SphereRegionStruct()` specifies a spherical mesh as the spherical region specified in the structure as defined in subclause 7.5.6. The `SphereRegionStruct()` shall be inferred to have shape type value equal to 1 (specified by two azimuth circles and two elevation circles). When `mesh_type` is equal to 2, the `SphereRegionStruct()` is inferred to have shape type value equal to 0 (specified by four great circles). The value of the `interpolate` syntax element for the `SphereRegionStruct()` is inferred to be equal to 0.

`3DParallelogramStruct()` specifies a parallelogram mesh in the 3D space. Two mesh elements described in this box shall not overlap.

`mesh_element_id` specifies an identifier to uniquely identify a mesh element within this mesh box. When `explicit_mesh_id_flag` is equal to 0, `mesh_element_id` for the i-th mesh element is inferred to be equal to `i`.

7.6.8.4 3D parallelogram structure

A `3DParallelogramStruct()` is a parallelogram in the local reference frame (O, X, Y, Z), as depicted in Figure 22. The vertices A, B, C, and D of the parallelogram, as depicted in Figure 22, are specified using the syntax elements `origin`, `u_direction`, and `v_direction` contained by the `3DParallelogramStruct()` as follows:

$$\begin{aligned} A &= O + \texttt{origin\_vertex} \\ B &= A + \texttt{u\_direction} \\ C &= B + \texttt{v\_direction} \\ D &= A + \texttt{v\_direction} \end{aligned} \qquad (18)$$

The **normal** vector of a `3DParallelogramStruct()` is pointing toward the direction of **uDir** × **vDir**, × being the cross product between two 3D vectors, **uDir** being the vector from A to B and **vDir** being the vector from A to D, as presented in Figure 22. The front face of the parallelogram is in the direction of the **normal**.

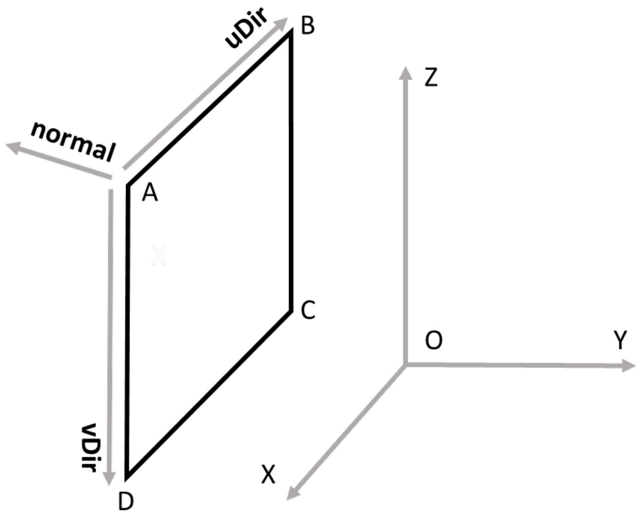


Figure 22 — 3D parallelogram

```
aligned(8) class 3DParallelogramStruct() {
   3DVectStruct origin_vertex;
   3DVectStruct u_direction;
   3DVectStruct v_direction;
}
```

`origin_vertex` is a vector used to specify the vertex A of the parallelogram.

`u_direction` is a vector used to specify the vertex B of the parallelogram.

`v_direction` is a vector used to specify the vertices C and D of the parallelogram.

7.6.8.5 3D vector structure

This structure specifies a vector in a 3D cartesian coordinate system. The vector coordinates are expressed in the local coordinate system.

```
aligned(8) class 3DVectStruct() {
   signed int(32) x_comp;
   signed int(32) y_comp;
   signed int(32) z_comp;
}
```

`x_comp`, `y_comp`, and `z_comp` represent, respectively, the x, y, and z cartesian coordinate of a 3D vector expressed in the local coordinate system. The value is relative to the unit sphere and is in units of 10^{-7}.

7.7 Timed metadata for sphere regions

7.7.1 General

Subclause 7.7 specifies a generic timed metadata track syntax for indicating sphere regions. The purpose for the timed metadata track is indicated by the track sample entry type. The sample format of all metadata tracks specified in subclause 7.7 starts with a common part and may be followed by an extension part that is specific to the sample entry of the metadata track. Each sample specifies a sphere region.

When a sphere region timed metadata track is linked to one or more media tracks with a 'cdsc' track reference, it describes each media track individually.

NOTE The syntax allows for one sample to specify multiple sphere regions. However, there is a semantic restriction that limits the samples to have only one sphere region.

7.7.2 Sample entry

7.7.2.1 Definition

Exactly one SphereRegionConfigBox shall be present in the sample entry. SphereRegionConfigBox specifies the shape of the sphere region specified by the samples. When the azimuth and elevation ranges of the sphere region in the samples do not change, they may be indicated in the sample entry.

7.7.2.2 Syntax

```
class SphereRegionSampleEntry(type) extends MetaDataSampleEntry(type) {
   SphereRegionConfigBox(); // mandatory
   Box[] other_boxes; // optional
}

class SphereRegionConfigBox extends FullBox('rosc', 0, 0) {
   unsigned int(8) shape_type;
   bit(7) reserved = 0;
   unsigned int(1) dynamic_range_flag;
   if (dynamic_range_flag == 0) {
      unsigned int(32) static_azimuth_range;
      unsigned int(32) static_elevation_range;
   }
   unsigned int(8) num_regions;
}
```

7.7.2.3 Semantics

shape_type equal to 0 specifies that the sphere region is specified by four great circles. shape_type equal to 1 specifies that the sphere region is specified by two azimuth circles and two elevation circles. shape_type values greater than 1 are reserved. The value of shape_type is used as the shape type value when applying subclause 7.5.6 to the semantics of the samples of the sphere region metadata track.

dynamic_range_flag equal to 0 specifies that the azimuth and elevation ranges of the sphere region remain unchanged in all samples referring to this sample entry. dynamic_range_flag equal to 1 specifies that the azimuth and elevation ranges of the sphere region are indicated in the sample format.

static_azimuth_range and static_elevation_range specify the azimuth and elevation ranges, respectively, of the sphere region for each sample referring to this sample entry in units of 2^{-16} degrees. static_azimuth_range and static_elevation_range specify the ranges through the centre point of the sphere region, as illustrated by Figure 20 or Figure 21. static_azimuth_range shall be in the range of 0 to 360 * 2^{16}, inclusive. static_elevation_range shall be in the range of 0 to 180 * 2^{16}, inclusive. When static_azimuth_range and static_elevation_range are present and are both equal to 0, the sphere region for each sample referring to this sample entry is a point on a spherical surface. When static_azimuth_range and static_elevation_range are present, the values of azimuth_range and elevation_range are inferred to be equal to static_azimuth_range and static_elevation_range, respectively, when applying subclause 7.5.6 to the semantics of the samples of the sphere region metadata track.

num_regions specifies the number of sphere regions in the samples referring to this sample entry. num_regions shall be equal to 1. Other values of num_regions are reserved.

7.7.3 Sample format

7.7.3.1 Definition

Each sample specifies a sphere region. The SphereRegionSample structure may be extended in derived track formats.

7.7.3.2 Syntax

```
aligned(8) SphereRegionSample() {
   for (i = 0; i < num_regions; i++)
      SphereRegionStruct(dynamic_range_flag, 1);
}
```

7.7.3.3 Semantics

Subclause 7.5.6 applies to the sample that contains the SphereRegionStruct() structure.

Let the target media samples be the media samples in the referenced media tracks with composition times greater than or equal to the composition time of this sample and less than the composition time of the next sample.

interpolate equal to 0 specifies that the values of centre_azimuth, centre_elevation, centre_tilt, azimuth_range (if present), and elevation_range (if present) in this sample apply to the target media samples. interpolate equal to 1 specifies that the values of centre_azimuth, centre_elevation, centre_tilt, azimuth_range (if present), and elevation_range (if present) that apply to the target media samples are linearly interpolated based on composition times, from the values of the corresponding fields in this sample and the previous sample.

The value of interpolate for a sync sample, the first sample of the track, and the first sample of a track fragment shall be equal to 0.

7.7.4 Initial viewing orientation

7.7.4.1 Definition

This metadata indicates initial viewing orientations that should be used when playing the associated media tracks or a single omnidirectional image stored as an image item.

When the playback of a media track that contains nothing but one or more overlay sources is intended to be started using another viewing orientation than that indicated by (centre_azimuth, centre_elevation, centre_tilt) equal to (0, 0, 0) relative to the global coordinate axes, an initial viewing orientation metadata track shall be present and associated with the media track. In the absence of this type of metadata centre_azimuth, centre_elevation, and centre_tilt should all be inferred to be equal to 0.

NOTE 1 When a media track consists of one or more overlay sources, it is consumed with background visual media, and the initial viewing orientation is present or inferred for the background visual media.

When playing a file and when the file contains an initial viewing orientation metadata track, OMAF players are expected to parse the initial viewing orientation metadata track associated with a media track and obey it when rendering the media track.

An OMAF player should use the indicated or inferred centre_azimuth, centre_elevation, and centre_tilt values as follows:

— If the orientation/viewport metadata of the OMAF player is obtained on the basis of an orientation sensor included in or attached to a viewing device, the OMAF player should

  — obey only the centre_azimuth value, and

  — ignore the values of centre_elevation and centre_tilt and use the respective values from the orientation sensor instead.

— Otherwise, the OMAF player should obey all three of centre_azimuth, centre_elevation, and centre_tilt.

The track sample entry type 'invo' shall be used.

shape_type shall be equal to 0, dynamic_range_flag shall be equal to 0, static_azimuth_range shall be equal to 0, and static_elevation_range shall be equal to 0 in the SphereRegionConfigBox of the sample entry.

NOTE 2 This metadata applies to any viewport regardless of which azimuth and elevation ranges are covered by the viewport. Thus, dynamic_range_flag, static_azimuth_range, and static_elevation_range do not affect the dimensions of the viewport that this metadata concerns and are hence required to be equal to 0. When the OMAF player obeys the centre_tilt value as concluded above, the value of centre_tilt can be interpreted by setting the azimuth and elevation ranges for the sphere region of the viewport equal to those that are actually used in displaying the viewport.

7.7.4.2 Sample syntax

```
class InitialViewingOrientationSample() extends SphereRegionSample() {
   unsigned int(1) refresh_flag;
   bit(7) reserved = 0;
}
```

7.7.4.3 Sample semantics

NOTE 1 As the sample structure extends from SphereRegionSample, the syntax elements of SphereRegionSample are included in the sample.

centre_azimuth, centre_elevation, and centre_tilt specify the viewing orientation in units of 2^{-16} degrees relative to the global coordinate axes. centre_azimuth and centre_elevation indicate the centre of the viewport, and centre_tilt indicates the tilt angle of the viewport.

interpolate shall be equal to 0.

refresh_flag equal to 0 specifies that the indicated viewing orientation should be used when starting the playback from a time-parallel sample in an associated media track. refresh_flag equal to 1 specifies that the indicated viewing orientation should always be used when rendering the time-parallel sample of each associated media track, i.e. both in continuous playback and when starting the playback from the time-parallel sample.

NOTE 2 refresh_flag equal to 1 enables the content author to indicate that a particular viewing orientation is recommended even when playing the video continuously. For example, refresh_flag equal to 1 can be indicated for a scene cut position.

7.7.5 Recommended viewport

7.7.5.1 Definition

The recommended viewport timed metadata track indicates the viewport that should be displayed when the user does not have control of the viewing orientation or has released control of the viewing orientation.

NOTE 1 The recommended viewport timed metadata track can be used for indicating a recommended viewport based on a director's cut or based on measurements of viewing statistics.

If the timed metadata track concerns more than one viewpoint or the referenced media tracks contain media data for more than one viewpoint, the track sample entry type `'rvp2'` shall be used. Otherwise, the track sample entry type `'rcvp'` or `'rvp2'` shall be used.

Each video track referenced by a `'cdsc'` track reference from a recommended viewport timed metadata track shall cover the indicated recommended viewports completely for the entire duration of the timed metadata track. The group of video tracks that are referenced by a `'cdtg'` track reference from a recommended viewport timed metadata track shall collectively cover the indicated recommended viewports completely for the entire duration of the timed metadata track.

NOTE 2 When a recommended viewport timed metadata track is used to derive the viewport and the video track(s) referenced by the recommended viewport timed metadata track have associated overlays, an OMAF player is expected to render the overlays like in rendering with user-controlled viewing orientation.

When playing the content by following a recommended viewport timed metadata track indicating a viewpoint switch, if the current viewpoint's `ViewpointInformationStruct()` contains `ViewpointSwitchingListStruct()`, an OMAF player should obey the timeline switching offsets and transition effects as specified in the current viewpoint's `ViewpointSwitchingListStruct()`. When there is more than one entry corresponding to the same `destination_viewpoint_id` in `ViewpointSwitchingListStruct()`, the first entry should be used.

The recommended viewport metadata track (`'rcvp'` or `'rvp2'`) may be linked to one or more tracks carrying the recommended-viewport rectangular 2D media content that it defines by means of an `'esri'` (encoded spherical region-of-interest) track reference from the recommended viewport metadata track to the rectangular 2D video track. Figure 23 illustrates the relationship between the recommended viewport and the rectangular 2D video, linked via the `'esri'` track reference.

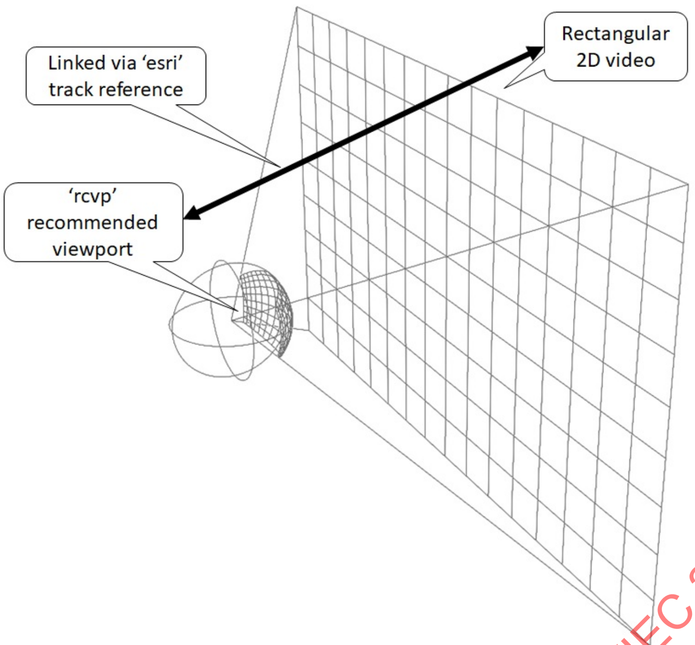

Linked via 'esri' track reference
Rectangular 2D video
'rcvp' recommended viewport

Figure 23 — Illustration of '`esri`' track reference

7.7.5.2 Sample entry syntax

```
class RcvpSampleEntry() extends SphereRegionSampleEntry('rcvp') {
   RcvpInfoBox(); // mandatory
}

class Rvp2SampleEntry() extends SphereRegionSampleEntry('rvp2') {
   RcvpInfoBox(); // mandatory
}

class RcvpInfoBox extends FullBox('rvif', version, 0) {
   unsigned int(8) viewport_type;
   string viewport_description;
   if (version > 0) {
      unsigned int(2) viewpoint_idc;
      bit(6) reserved = 0;
      if(viewpoint_idc == 1)
         unsigned int(32) rvif_viewpoint_id;
   }
}
```

7.7.5.3 Sample entry semantics

`version` shall be equal to 0 when `RcvpInfoBox` is contained in a `'rcvp'` sample entry. `version` shall be equal to 1 when `RcvpInfoBox` is contained in `'rvp2'` sample entry.

`viewport_type` specifies the type of the recommended viewport as listed in Table 13.

Table 13 — Recommended viewport type

Value	Description
0	A recommended viewport per the director's cut, i.e. a viewport suggested according to the creative intent of the content author or content provider
1	A recommended viewport selected based on measurements of viewing statistics
2..239	Reserved (for use by future extensions of ISO/IEC 23090-2)
240..255	Unspecified (for use by applications or external specifications)

`viewport_description` is null-terminated UTF-8 string that provides a textual description of the recommended viewport.

`viewpoint_idc` equal to 0 specifies that all the media tracks referenced by this timed metadata track represent the same viewpoint. `viewpoint_idc` equal to 1 specifies that the viewpoint identifiers referenced by the sample entry containing this `RcvpInfoBox` represent the viewpoint with viewpoint identifier equal to `rvif_viewpoint_id`. `viewpoint_idc` equal to 2 specifies that the samples contain `viewpoint_id`. `viewpoint_idc` equal to 3 is reserved. When not present, `viewpoint_idc` is inferred to be equal to 0.

`rvif_viewpoint_id` specifies the viewpoint identifier that identifies the viewpoint containing the recommended viewport for the samples referencing the sample entry containing this `RcvpInfoBox`.

7.7.5.4 Sample syntax

```
class RecommendedViewportSample() extends SphereRegionSample() {
   if (viewpoint_idc == 2)
      unsigned int(32) viewpoint_id;
}
```

7.7.5.5 Sample semantics

`shape_type` shall be equal to 0 in the `SphereRegionConfigBox` of the sample entry.

`static_azimuth_range` and `static_elevation_range`, when present, or `azimuth_range` and `elevation_range`, when present, indicate the azimuth and elevation ranges, respectively, of the recommended viewport.

`centre_azimuth` and `centre_elevation` indicate the centre point of the recommended viewport relative to the global coordinate axes. `centre_tilt` indicates the tilt angle of the recommended viewport.

`viewpoint_id` specifies the viewpoint identifier of the viewpoint that contains the recommended viewport.

7.7.6 Timed text sphere location metadata

7.7.6.1 General

The timed text sphere location metadata indicates the timed text sphere location that is used, together with other information, to determine where the timed text is placed and displayed in 3D space. Since the timed text cues are rendered at certain positions in 3D space, they are only visible when timed text sphere location is within the sphere region defining the viewport.

7.7.6.2 Sample entry format

The track sample entry type `'ttsl'` shall be used.

The sample entry of this sample entry type is specified as follows:

```
class TTSphereLocationSampleEntry() extends SphereRegionSampleEntry('ttsl') {
   unsigned int(1) depth_included_flag;
   bit(7) reserved = 0;
}
```

`depth_included_flag` equal to 1 specifies that the depth (z-value) of the regions on which the timed text is to be rendered is present in the samples. The value 0 specifies that the depth (z-value) of the regions on which the timed text is to be rendered is not present in the samples.

When `SphereRegionSampleEntry()` is included in `TTSphereLocationSampleEntry()`, the following applies:

The values of `shape_type`, `dynamic_range_flag`, `static_azimuth_range`, and `static_elevation_range` shall all be equal to 0.

The value of `num_regions` may be greater than 1.

7.7.6.3 Sample format

The sample format of timed text sphere location timed metadata is specified as follows:

```
aligned(8) TTSphereLocationSample() extends SphereRegionSample() {
   for (i=0; i<num_regions; i++) {
      string region_id;
      if (depth_included_flag)
         unsigned int(16) region_depth;
   }
}
```

`region_id` provides the identifier of the region on which the timed text is to be rendered. `region_id` should be equal to the identification of the corresponding region defined in the timed text streams in the IMSC1 or WebVTT track.

`region_depth` indicates the depth (z-value) of the region on which the timed text is to be rendered. The depth value is the norm of the normal vector of the timed text region. This value is relative to a unit sphere and is in units of 2^{-16}.

When `SphereRegionStruct()` is included in the `TTSphereLocationSample()` structure, it indicates the timed text sphere location that is used, together with other information, to determine where the timed text is placed and displayed in 3D space.

7.8 Signalling of region-wise quality ranking

7.8.1 General

Quality ranking values of quality ranking regions relative to other quality ranking regions of the same track or quality ranking regions of other tracks may be indicated by using the `SphereRegionQualityRankingBox` or the `2DRegionQualityRankingBox`. When neither `SphereRegionQualityRankingBox` nor `2DRegionQualityRankingBox` is present in a visual sample entry, the quality ranking value for the visual track is not defined. Quality ranking values indicate a relative quality order of quality ranking regions. When quality ranking region A has a non-zero quality ranking value less than that of quality ranking region B, quality ranking region A has a higher quality than quality ranking region B. When the quality ranking value is non-zero, the picture quality within the entire indicated quality ranking region is approximately constant. The boundaries of the quality ranking sphere regions specified by the `SphereRegionQualityRankingBox` may or may not match with the boundaries of the quality ranking 2D regions specified by the `2DRegionQualityRankingBox`. The boundaries of the quality ranking sphere or 2D regions may or may not match with the boundaries of the packed regions or the boundaries of the projected regions specified by `RegionWisePackingBox`.

Either or both of `SphereRegionQualityRankingBox` and the `2DRegionQualityRankingBox` should be present for the projected omnidirectional video tracks to enable viewport-dependent content selection.

7.8.2 Spherical region-wise quality ranking

7.8.2.1 Definition

Box type:	`'srqr'`
Container:	`VisualSampleEntry`
Mandatory (per an item):	No
Quantity (per an item):	At most one for each `region_definition_type` value

7.8.2.2 Syntax

```
aligned(8) class SphereRegionQualityRankingBox extends FullBox('srqr', 0, 0) {
   SphereRegionQualityRankingStruct();
}

aligned(8) class SphereRegionQualityRankingStruct() {
   unsigned int(8) region_definition_type;
   unsigned int(8) num_regions;
   unsigned int(1) remaining_area_flag;
   unsigned int(1) view_idc_presence_flag;
   unsigned int(1) quality_ranking_local_flag;
   unsigned int(4) quality_type;
   bit(1) reserved = 0;
   if (view_idc_presence_flag == 0) {
      unsigned int(2) default_view_idc;
      bit(6) reserved = 0;
   }
   for (i = 0; i < num_regions; i++) {
      unsigned int(8) quality_ranking;
      if (view_idc_presence_flag == 1) {
         unsigned int(2) view_idc;
         bit(6) reserved = 0;
      }
      if (quality_type == 1) {
         unsigned int(16) orig_width;
         unsigned int(16) orig_height;
      }
      if ((i < (num_regions - 1)) || (remaining_area_flag == 0))
         SphereRegionStruct(1, 1);
   }
}
```

7.8.2.3 Semantics

region_definition_type has identical semantics to shape_type of SphereRegionConfigBox.

num_regions specifies the number of quality ranking sphere regions for which the quality ranking information is given in this box. Value 0 is reserved. There shall be no point on the sphere that is contained in more than one of these quality ranking sphere regions.

remaining_area_flag equal to 0 specifies that all the quality ranking sphere regions are defined by the SphereRegionStruct(1, 1) structures. remaining_area_flag equal to 1 specifies that the first num_regions - 1 quality ranking sphere regions are defined by SphereRegionStruct(1, 1) structure and the last remaining quality ranking sphere region is the sphere region within the content coverage, not covered by the union of the quality ranking sphere regions defined by the first num_regions - 1 SphereRegionStruct(1, 1) structures. The last remaining quality ranking sphere region may be on both the left and right views.

view_idc_presence_flag equal to 0 specifies that view_idc is not present. view_idc_presence_flag equal to 1 specifies that view_idc is present and indicates the association of quality ranking sphere region with particular (left or right or both) views or monoscopic content.

quality_ranking_local_flag equal to 1 specifies that the quality ranking information provided in this instance of this box is relative to the quality ranking information provided in all instances of this box for this track only. quality_ranking_local_flag equal to 0 specifies that the quality ranking information provided in this instance of this box is relative to the quality ranking information provided in all instances of SphereRegionQualityRankingBox and 2DRegionQualityRankingBox with quality_ranking_local_flag equal to 0 for this and other tracks.

NOTE 1 quality_ranking_local_flag equal to 1 can be used in a track that includes coded video data by reference to a set of tracks among which one is selected as the source of the coded video data. For example, an 'alte' track reference specified in ISO/IEC 14496-15 can be used for this purpose. SphereRegionQualityRankingBox or 2DQualityRankingBox with quality_ranking_local_flag equal to 0 can be present in the referenced tracks within the file.

quality_type indicates which factor causes the differences in the quality of packed regions on the picture. quality_type equal to 0 specifies that all packed regions correspond to the same projected picture resolution. quality_type equal to 1 specifies that at least one horRatio value, as derived in subclause 5.4.2, may differ from other horRatio values among all pairs of packed and projected regions of the picture or at least one verRatio value, as derived in subclause 5.4.2, may differ from other verRatio values among all pairs of packed and projected regions of the picture. quality_type values greater than 1 are reserved.

default_view_idc equal to 0 indicates that the quality ranking sphere region is monoscopic, 1 indicates that the quality ranking sphere region is on the left view of stereoscopic content, 2 indicates that the quality ranking sphere region is on the right view of stereoscopic content, 3 indicates that the quality ranking sphere region is on both the left and right views.

quality_ranking specifies a quality ranking value of the quality ranking sphere region. quality_ranking equal to 0 indicates that the quality ranking value is not defined. The semantics of non-zero quality ranking values are specified in subclause 7.8.1.

view_idc equal to 0 indicates that the quality ranking sphere region is monoscopic, 1 indicates that the quality ranking sphere region is on the left view of stereoscopic content, 2 indicates that the quality ranking sphere region is on the right view of stereoscopic content, 3 indicates that the quality ranking sphere region is on both the left and right views. When not present, the value of `view_idc` is inferred to be equal to the value of `default_view_idc`.

`orig_width` and `orig_height` specify the width and height, respectively, of such a monoscopic projected picture for which both horRatio and verRatio, as derived in subclause 5.4.2 for each of the packed regions that cover the quality ranking sphere region, are equal to 1.

NOTE 2 `orig_width` and `orig_height` represent the width and height of the picture from which the packed region has been extracted without resampling.

NOTE 3 A player is suggested to parse `SphereRegionQualityRankingBox` and select the track for playing that matches the user's viewing orientation in a manner that:

— The quality ranking value on the region covering the viewport is greater than 0 and less than that for other regions.

— The resolution of the region covering the viewport is suitable for the display. If `quality_type` is equal to 1, `orig_width` and `orig_height` represent the width and height of the monoscopic projected picture from which the packed region covering the viewport has been extracted. Otherwise, `width` and `height` of `VisualSampleEntry` can be used to conclude the resolution on the viewport.

`SphereRegionStruct(1, 1)` specifies the spherical location and size of the quality ranking sphere region relative to the global coordinate axes, while the shape type value of the quality ranking sphere regions is indicated by `region_definition_type`. The value of `interpolate` in `SphereRegionStruct(1, 1)` shall be equal to 0.

7.8.3 2D region-wise quality ranking

7.8.3.1 Definition

Box type:	`'2dqr'`
Container:	`VisualSampleEntry`
Mandatory (per an item):	No
Quantity (per an item):	Zero or one

7.8.3.2 Syntax

```
aligned(8) class 2DRegionQualityRankingBox extends FullBox('2dqr', 0, 0) {
   2DRegionQualityRankingStruct();
}

aligned(8) class 2DRegionQualityRankingStruct() {
   unsigned int(8) num_regions;
   unsigned int(1) remaining_area_flag;
   unsigned int(1) view_idc_presence_flag;
   unsigned int(1) quality_ranking_local_flag;
   unsigned int(4) quality_type;
   bit(1) reserved = 0;
   if (view_idc_presence_flag == 0) {
      unsigned int(2) default_view_idc;
      bit(6) reserved = 0;
   }
   for (i = 0; i < num_regions; i++) {
      unsigned int(8) quality_ranking;
      if (view_idc_presence_flag == 1) {
         unsigned int(2) view_idc;
         bit(6) reserved = 0;
      }
      if (quality_type == 1) {
            unsigned int(16) orig_width;
            unsigned int(16) orig_height;
         }
         if ((i < (num_regions - 1)) || (remaining_area_flag == 0)) {
            unsigned int(16) left_offset;
            unsigned int(16) top_offset;
            unsigned int(16) region_width;
            unsigned int(16) region_height;
         }
      }
   }
```

7.8.3.3 Semantics

quality_ranking, view_idc_presence_flag, default_view_idc, and view_idc are specified identically to the syntax elements with the same names in SphereRegionQualityRankingBox but are further constrained as follows:

— When a 2DRegionQualityRankingBox is present in a VisualSampleEntry and StereoVideoBox is present in the VisualSampleEntry and indicates temporal interleaving, view_idc_presence_flag shall be equal to 0 and default_view_idc shall be equal to 3.

num_regions specifies the number of quality ranking 2D regions for which the quality ranking information is given in this box. Value 0 is reserved. There shall be no pixel of the decoded picture that is contained in more than one of these quality ranking 2D regions.

remaining_area_flag equal to 0 specifies that all the quality ranking 2D regions are defined by the left_offset, top_offset, region_width, and region_height. remaining_area_flag equal to 1 specifies that the first num_regions − 1 quality ranking 2D regions are defined by left_offset, top_offset, region_width, and region_height and the last remaining quality ranking 2D region is the area in the picture with width equal to width of VisualSampleEntry and height equal to height of VisualSampleEntry, not covered by the union of the first num_regions − 1 quality ranking 2D regions. The last remaining quality ranking 2D region may be on both the left and right views.

quality_ranking_local_flag equal to 1 specifies that the quality ranking information provided in this instance of this box is relative to the quality ranking information provided in all instances of this box for this track only. quality_ranking_local_flag equal to 0 specifies that the quality ranking information provided in this instance of this box is relative to the quality ranking information provided in all instances of SphereRegionQualityRankingBox and 2DRegionQualityRankingBox with quality_ranking_local_flag equal to 0 for this and other tracks.

NOTE 1 quality_ranking_local_flag equal to 1 can be used in a track that includes coded video data by reference to a set of tracks among which one is selected as the source of the coded video data. For example, an 'alte' track reference specified in ISO/IEC 14496-15 can be used for this purpose. SphereRegionQualityRankingBox or 2DQualityRankingBox with quality_ranking_local_flag equal to 0 can be present in the referenced tracks within the file.

quality_type indicates which factor causes the differences in the quality of packed regions on the picture. quality_type equal to 0 specifies that all packed regions correspond to the same projected picture resolution. quality_type equal to 1 specifies that at least one horRatio value, as derived in subclause 5.4.2, may differ from other horRatio values among all pairs of packed and projected regions of the picture or at least one verRatio value, as derived in subclause 5.4.2, may differ from other verRatio values among all pairs of packed and projected regions of the picture. quality_type values greater than 1 are reserved.

orig_width and orig_height specify the width and height, respectively, of such a monoscopic projected picture for which both horRatio and verRatio, as derived in subclause 5.4.2 for each of the packed regions that cover the quality ranking 2D region, are equal to 1.

NOTE 2 orig_width and orig_height represent the width and height of the picture from which the packed region has been extracted without resampling.

NOTE 3 A player is suggested to parse 2DRegionQualityRankingBox and select the track for playing that matches the user's viewing orientation in a manner that:

— The quality ranking value on the region covering the viewport is greater than 0 and less than that for other regions.

— The resolution of the region covering the viewport is suitable for the display. If quality_type is equal to 1, orig_width and orig_height represent the width and height of the monoscopic projected picture from which the packed region covering the viewport has been extracted. Otherwise, width and height of VisualSampleEntry can be used to conclude the resolution on the viewport.

left_offset, top_offset, region_width, and region_height are integer values that indicate the position and size of the quality ranking 2D region. left_offset and top_offset indicate the horizontal and vertical coordinates, respectively, of the upper left corner of the quality ranking 2D region within the picture, in units of luma samples. region_width and region_height indicate the width and height, respectively, of the quality ranking 2D region within the picture, in units of luma samples. left_offset + region_width shall be less than width of VisualSampleEntry. top_offset + region_height shall be less than height of VisualSampleEntry.

region_width shall be greater than 0.

region_height shall be greater than 0.

7.9 Storage of omnidirectional images

7.9.1 General

Omnidirectional images are stored in a file as image items, as specified in ISO/IEC 23008-12. The ProjectionFormatProperty shall be present for an omnidirectional image item. When an image item contains stereoscopic content, the FramePackingProperty shall be present for the image item. When an image item contains a packed picture generated by region-wise packing from the projected picture, the RegionWisePackingProperty shall be present for the image item.

7.9.2 Frame packing item property

7.9.2.1 Definition

Box type:	'stvi'
Property type:	Descriptive item property
Container:	ItemPropertyContainerBox
Mandatory (per an item):	No
Quantity (per an item):	Zero or one

FramePackingProperty indicates that the reconstructed image contains a representation of two spatially packed constituent pictures.

essential shall be equal to 1 for a 'stvi' item property.

7.9.2.2 Syntax

FramePackingProperty has the same syntax as StereoVideoBox specified in ISO/IEC 14496-12.

7.9.2.3 Semantics

The semantics of the syntax elements within the FramePackingProperty are the same as those specified for the syntax elements of StereoVideoBox as defined in ISO/IEC 14496-12 and in subclause 7.1.3.

7.9.3 Projection format item property

7.9.3.1 Definition

Box type:	'prfr'
Property type:	Descriptive item property
Container:	ItemPropertyContainerBox
Mandatory (per an item):	No
Quantity (per an item):	Zero or one

ProjectionFormatProperty indicates the omnidirectional projection format of the image.

When 'prfr' is present, the reconstructed image represents a packed picture that has been generated as indicated in Figure 2 and Figure 3 for a monoscopic and stereoscopic image, respectively. The semantics of the sample locations of the reconstructed image are specified in subclause 7.5.1.2.

The format of the projected monoscopic pictures is indicated with the ProjectionFormatProperty.

For stereoscopic video, the frame packing arrangement of the projected left and right pictures is indicated with the FramePackingProperty. The absence of FramePackingProperty indicates that the content of the image item is monoscopic.

When both ProjectionFormatProperty and FramePackingProperty are present for an image item, stereo_scheme shall be equal to 4, and the first byte of stereo_indication_type shall be equal to 3 or 4, and the second byte of stereo_indication_type shall be equal to 0 indicating that quincunx sampling is not in use.

Optional region-wise packing is indicated with the RegionWisePackingProperty. The absence of RegionWisePackingProperty indicates that no region-wise packing is applied.

essential shall be equal to 1 for a 'prfr' item property.

The values of the variables HorDiv1 and VerDiv1 are set as follows for an image item:

— If FramePackingProperty is not present for the image item, HorDiv1 is set equal to 1 and VerDiv1 is set equal to 1.

— Otherwise (FramePackingProperty is present for the image item), the following applies:

  — If side-by-side frame packing is indicated, HorDiv1 is set equal to 2 and VerDiv1 is set equal to 1.

  — Otherwise (top-bottom frame packing is indicated), HorDiv1 is set equal to 1 and VerDiv1 is set equal to 2.

The width and height of a monoscopic projected luma picture (ConstituentPicWidth and ConstituentPicHeight, respectively) corresponding to the reconstructed image of the image item are derived as follows:

— If RegionWisePackingProperty is not present for the image item, ConstituentPicWidth and ConstituentPicHeight are set to be equal to image_width / HorDiv1 and image_height / VerDiv1, respectively, where image_width and image_height are syntax elements of ImageSpatialExtentsProperty associated with the image item.

— Otherwise, ConstituentPicWidth and ConstituentPicHeight are set equal to proj_picture_width / HorDiv1 and proj_picture_height / VerDiv1, respectively.

If RotationProperty is not present for the image item, RotationFlag is set equal to 0. Otherwise, RotationFlag is set equal to 1.

If FramePackingProperty is not present for the image item, SpatiallyPackedStereoFlag, TopBottomFlag, and SideBySideFlag are set equal to 0. Otherwise, the following applies:

— When the FramePackingProperty indicates top-bottom frame packing, SpatiallyPackedStereoFlag is set equal to 1, TopBottomFlag is set equal to 1, and SideBySideFlag is set equal to 0.

— When the FramePackingProperty indicates side-by-side frame packing, SpatiallyPackedStereoFlag is set equal to 1, TopBottomFlag is set equal to 0, and SideBySideFlag is set equal to 1.

If RegionWisePackingProperty is not present for the image item, RegionWisePackingFlag is set equal to 0. Otherwise, RegionWisePackingFlag is set equal to 1.

7.9.3.2 Syntax

```
aligned(8) class ProjectionFormatProperty
extends ItemFullProperty('prfr', 0, 0) {
   ProjectionFormatStruct(); /* specified in subclause 7.5.2 */
}
```

7.9.4 Essential fisheye image item property

7.9.4.1 General

When an image item contains a picture which consists of multiple circular images captured by fisheye cameras, the EssentialFisheyeImageProperty shall be present for the image item.

7.9.4.2 Definition

Box type:	'fovi'
Property type:	Descriptive item property
Container:	ItemPropertyContainerBox
Mandatory (per an item):	No
Quantity (per an item):	Zero or one

EssentialFisheyeImageProperty provides essential fisheye parameters for stitching and rendering fisheye images at the OMAF player. The fields in EssentialFisheyeImageProperty provide region information of circular images in the image item and the field of view and camera parameters of the fisheye lens(es).

When the 'fovi' item property is present, the 'prfr' item property shall not be present.

essential shall be equal to 1 for a 'fovi' item property.

7.9.4.3 Syntax

```
aligned(8) class EssentialFisheyeImageProperty
extends ItemFullProperty('fovi', 0, 0) {
   FisheyeVideoEssentialInfoStruct(); /*specified in Clause 6*/
}
```

7.9.4.4 Semantics

The semantics are as specified in subclause 6.2.2.

7.9.5 Supplemental fisheye image item property

7.9.5.1 Definition

Box type:	`'fvsi'`
Property type:	Descriptive item property
Container:	`ItemPropertyContainerBox`
Mandatory (per an item):	No
Quantity (per an item):	Zero or one

`SupplementalFisheyeImageProperty` provides supplemental fisheye parameters, such as the local field of view information for high quality stitching and rendering as well as deadzone information.

When the `'fvsi'` item property is present, the `'fovi'` item property shall be present.

7.9.5.2 Syntax

```
aligned(8) class SupplementalFisheyeImageProperty
extends ItemFullProperty('fvsi', version, 0) {
   FisheyeVideoSupplementalInfoStruct(version); /*specified in Clause 6*/
}
```

7.9.5.3 Semantics

The semantics are as specified in subclause 6.3.2.

7.9.6 Region-wise packing item property

7.9.6.1 Definition

Box type:	`'rwpk'`
Property type:	Descriptive item property
Container:	`ItemPropertyContainerBox`
Mandatory (per an item):	No
Quantity (per an item):	Zero or one

`RegionWisePackingProperty` specifies the mapping between packed regions and the corresponding projected regions and specifies the location and size of the guard bands, if any.

`essential` shall be equal to 1 for a `'rwpk'` item property.

7.9.6.2 Syntax

```
aligned(8) class RegionWisePackingProperty
extends ItemFullProperty('rwpk', 0, 0) {
   RegionWisePackingStruct(); /* specified in subclause 7.5.3 */
}
```

7.9.6.3 Semantics

Subclause 7.5.3 applies with the following additional constraint:

— packed_picture_width and packed_picture_height shall have such values that packed_picture_width is an integer multiple of image_width and packed_picture_height is an integer multiple of image_height, where image_width and image_height are syntax elements of the ImageSpatialExtentsProperty associated to the image item.

7.9.7 Rotation item property

7.9.7.1 General

When an image item contains an omnidirectional image specified based on the local coordinate axes, the RotationProperty shall be present for the image item.

7.9.7.2 Definition

Box type:	'rotn'
Property type:	Descriptive item property
Container:	ItemPropertyContainerBox
Mandatory (per an item):	No
Quantity (per an item):	Zero or one

RotationProperty is used to indicate the yaw, pitch, and roll angles, respectively, of the rotation to be applied to convert the local coordinate axes to the global coordinate axes. In the case of stereoscopic omnidirectional image, the fields apply to each view individually. The absence of RotationProperty indicates that rotation_yaw, rotation_pitch, and rotation_roll are all inferred to be equal to 0.

essential shall be equal to 1 for a 'rotn' item property.

7.9.7.3 Syntax

```
aligned(8) class RotationProperty
extends ItemFullProperty('rotn', 0, 0) {
   RotationStruct(); /* specified in subclause 7.5.4 */
}
```

7.9.8 Coverage information item property

7.9.8.1 General

7.9.8.2 Definition

Box type:	'covi'
Property type:	Descriptive item property
Container:	ItemPropertyContainerBox
Mandatory (per an item):	No
Quantity (per an item):	Zero or one

CoverageInformationProperty is used to indicate the content coverage of the omnidirectional image.

NOTE It is totally up to the OMAF player to handle the area that is not covered by the content when rendering the omnidirectional image content.

Each sphere location within the sphere regions specifying the content coverage shall have a corresponding sample in the reconstructed image. However, there may be some sphere locations that do have corresponding samples in the reconstructed image but are outside the content coverage.

7.9.8.3 Syntax

```
aligned(8) class CoverageInformationProperty
extends ItemFullProperty('covi', 0, 0) {
   ContentCoverageStruct()
}
```

7.9.9 Initial viewing orientation item property

7.9.9.1 Definition

Box type:	`'iivo'`
Property type:	Descriptive item property
Container:	`ItemPropertyContainerBox`
Mandatory (per an item):	No
Quantity (per an item):	Zero or one

`InitialViewingOrientationProperty` indicates the initial viewing orientation according to which the image should be initially rendered to the user.

When the viewing of the image item is intended to be started using another viewing orientation than that indicated by (`centre_azimuth`, `centre_elevation`, `centre_tilt`) equal to (0, 0, 0) relative to the global coordinate axes, an initial viewing orientation item property shall be present and associated with the image item. In the absence of this property `centre_azimuth`, `centre_elevation`, and `centre_tilt` should all be inferred to be equal to 0.

When an image item is associated with the initial viewing orientation item property, OMAF players are expected to parse the initial viewing orientation item property and obey it when viewing the item.

An OMAF player should use the indicated or inferred `centre_azimuth`, `centre_elevation`, and `centre_tilt` values as follows:

— If the orientation/viewport metadata of the OMAF player is obtained on the basis of an orientation sensor included in or attached to a viewing device, the OMAF player should

  — obey only the `centre_azimuth` value, and

  — ignore the values of `centre_elevation` and `centre_tilt` and use the respective values from the orientation sensor instead.

— Otherwise, the OMAF player should obey all three of `centre_azimuth`, `centre_elevation`, and `centre_tilt`.

For the syntax and semantics of `InitialViewingOrientationProperty`, `num_regions` is inferred to be equal to 1, `shape_type` is inferred to be equal to 0, `dynamic_range_flag` is inferred to be equal to 0, `static_azimuth_range` is inferred to be equal to 0, and `static_elevation_range` is inferred to be equal to 0.

NOTE This metadata applies to any viewport regardless of which azimuth and elevation ranges are covered by the viewport. Thus, `dynamic_range_flag`, `static_azimuth_range`, and `static_elevation_range` do not affect the dimensions of the viewport that this metadata concerns and are hence required to be equal to 0. When the OMAF player obeys the `centre_tilt` value as concluded above, the `centre_tilt` value can be interpreted by setting the azimuth and elevation ranges for the sphere region of the viewport equal to those that are actually used in displaying the viewport.

7.9.9.2 Syntax

```
aligned(8) class InitialViewingOrientationProperty
extends ItemFullProperty('iivo', 0, 0) {
   InitialViewingOrientationSample vr_initial_orientation;
}
```

7.9.9.3 Semantics

The semantics of the `InitialViewingOrientationSample` structure are specified in subclause 7.7.4.3 and apply to the image item by replacing the phrase "sample" with the phrase "image item" except that the semantics of `refresh_flag` are undefined and the flag shall be equal to 0.

7.10 Storage of timed text for omnidirectional video

7.10.1 General

Timed text is used for providing subtitles and closed captions for omnidirectional video. Timed text cues may be rendered on a certain region relative to the sphere (i.e. only visible when the user looks in a specific direction), or it may be rendered in a region on the current viewport (i.e. always visible irrespective of the viewing direction), in which case the text/cue region positions are relative to the current viewport.

For the timed text of the omnidirectional video, W3C Recommendation, *TTML Profiles for Internet Media Subtitles and Captions 1.0.1 (IMSC1)* or W3C Candidate Recommendation, *WebVTT, The Web Video Text Tracks Format* shall be used.

The IMSC1 or WebVTT streams shall be carried in tracks conforming to ISO/IEC 14496-30.

`XMLSubtitleSampleEntry` or `WVTTSampleEntry` shall include the `OmafTimedTextConfigBox` as specified in subclause 7.10.2.

7.10.2 OMAF timed text configuration box

7.10.2.1 Definition

Box Type:	`'otcf'`
Container:	`XMLSubtitleSampleEntry` or `WVTTSampleEntry`
Mandatory:	Yes (for timed text tracks associated with an omnidirectional video track)
Quantity:	One (for timed text tracks associated with an omnidirectional video track)

This box provides configuration information for presenting timed text together with omnidirectional video.

7.10.2.2 Syntax

```
aligned(8) class OmafTimedTextConfigBox extends FullBox('otcf', 0, 0) {
   unsigned int(1) relative_to_viewport_flag;
   unsigned int(1) relative_disparity_flag;
   unsigned int(1) depth_included_flag;
   bit(5) reserved = 0;
   unsigned int(8) region_count;
   for (i=0;i<region_count;i++) {
      string region_id;
      if(relative_to_viewport_flag == 1) {
         if (relative_disparity_flag)
            signed int(16) disparity_in_percent;
         else
            signed int(16) disparity_in_pixels;
      } else {
         SphereRegionStruct(0, 1);
         if (depth_included_flag)
            unsigned int(16) region_depth;
      }
   }
}
```

7.10.2.3 Semantics

relative_to_viewport_flag specifies how the timed text cues are to be rendered. The value 1 indicates that the timed text is expected to be always present on the display screen, i.e. the text cue is visible independently of the viewing direction of the user. The value 0 indicates that the timed text is expected to be rendered at a certain position on the sphere, i.e. the text cue is only visible when the user is looking in the direction where the text cue is rendered.

NOTE 1 When relative_to_viewport_flag is equal to 1, the active area where the timed text can be displayed is provided by the timed text track as a rectangular region.

relative_disparity_flag indicates whether the disparity is provided as a percentage value of the width of the display window for one view (when the value is equal to 1) or as a number of pixels (when the value is equal to 0).

depth_included_flag equal to 1 indicates that the depth (z-value) of regions on which the timed text is to be rendered is present. The value 0 indicates that the depth (z-value) of regions on which the timed text is to be rendered is not present.

region_count specifies the number of text regions for which a placement inside the sphere is provided. Each region is identified by an identifier. (both WebVTT and TTML identify regions using a unique ID). When a timed metadata track containing the timed text sphere metadata track is present and linked to this timed text track by the track reference of type 'cdsc', the value of region_count shall be 0.

NOTE 2 Both WebVTT and TTML identify a region using a unique identifier.

region_id provides the identifier of the text region. This identifier shall be equal to the identifier of the corresponding region defined in timed text streams in the IMSC1 or WebVTT track.

disparity_in_percent indicates the disparity, in units of 2^{-16}, as a fraction of the width of the display window for one view. The value may be negative, in which case the displacement direction is reversed. This value is used to displace the region to the left on the left eye view and to the right on the right eye view.

disparity_in_pixels indicates the disparity in pixels. The value may be negative, in which case the displacement direction is reversed. This value is used to displace the region to the left on the left eye view and to the right on the right eye view.

SphereRegionStruct() indicates a sphere location that is used, together with other information, to determine where the timed text is placed and displayed in 3D space. The vector between the centre of the sphere and this sphere location is the normal vector to the rendering 3D plane on which the timed text cue is to be rendered. This information and the depth of the 3D plane are used to determine the position of the rendering 3D plane in 3D space on which the timed text cue is to be rendered. An example of such region is shown in Figure 24 where the purple plane indicates the region and the line segment in black indicates the depth.

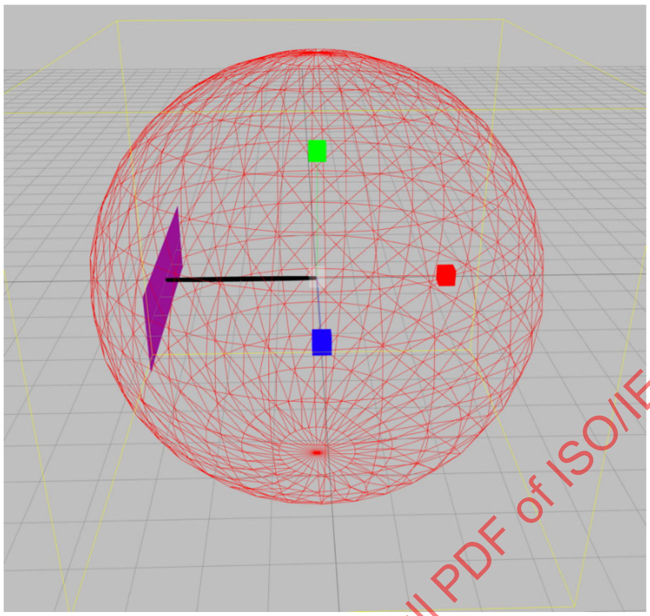


Figure 24 — An example of a region on which the timed text is rendered.

When SphereRegionStruct() is included in the OmafTimedTextConfigBox, the following applies:

For the syntax and semantics of SphereRegionStruct() included in the OmafTimedTextConfigBox, the values of shape_type, dynamic_range_flag, static_azimuth_range, and static_elevation_range are all inferred to be equal to 0.

For the SphereRegionStruct(0, 1) included in the OmafTimedTextConfigBox, interpolate shall be equal to 0.

centre_azimuth and centre_elevation specify the sphere location that is used, together with other information, to determine where the timed text is placed and displayed in 3D space. centre_azimuth shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive. centre_elevation shall be in the range of $-90 * 2^{16}$ to $90 * 2^{16}$, inclusive.

centre_tilt shall be equal to 0.

region_depth indicates the depth (z-value) of the region on which the timed text is to be rendered. The depth value is the norm of the normal vector of the timed text region. This value is relative to a unit sphere and is in units of 2^{-16}.

7.10.3 IMSC1 tracks

When IMSC1 streams are used, OmafTimedTextConfigBox shall be present in XMLSubtitleSampleEntry. When the disparity information is present in OmafTimedTextConfigBox, it is superseded by the tts:disparity attribute of the IMSC1 track.

```
class XMLSubtitleSampleEntry() extends SubtitleSampleEntry ('stpp') {
   string   namespace;
   string   schema_location;  // optional
   string   auxiliary_mime_types;
                  // optional, required if auxiliary resources are present
   OmafTimedTextConfigBox();         // optional
}
```

The `namespace` field of the `XMLSubtitleSampleEntry` shall contain one instance of the string `"http://www.w3.org/ns/ttml"`.

The `schema_location` field of the `XMLSubtitleSampleEntry` shall contain one of following string instances:

— `"http://www.w3.org/ns/ttml/profile/imsc1/text"`

— `"http://www.w3.org/ns/ttml/profile/imsc1/image"`

The `XMLSubtitleSampleEntry` shall contain a `MIMEBox` and its `content_type` field shall be constrained as follows:

— The `type` shall be `"application"`.

— The `subtype` shall be `"ttml+xml"`.

— The `codecs` parameter shall contain either `"im1t"` or `"im1i"` that indicates that an IMSC1 Text or Image processor is required, respectively.

All samples of the IMSC1 track shall conform to the Text Profile or Image Profile specified in W3C Recommendation, *TTML Profiles for Internet Media Subtitles and Captions 1.0.1 (IMSC1)*.

All IMSC1 sample format shall conform to ISO/IEC 14496-30.

The media type of the IMSC1 samples is `"application/mp4"`.

7.10.4 WebVTT tracks

When WebVTT is used, `OmafTimedTextConfigBox` shall be present in `WVTTSampleEntry`.

```
class WVTTSampleEntry() extends PlainTextSampleEntry ('wvtt') {
   WebVTTConfigurationBox  config;
   WebVTTSourceLabelBox    label;   // recommended
   OmafTimedTextConfigBox();        // optional
}
```

All WebVTT samples shall conform to the sample format as defined in ISO/IEC 14496-30:2018, subclause 6.6.

The media type of the WebVTT samples is `"text/vtt"`.

7.11 ERP region timed metadata

7.11.1 General

An ERP region timed metadata track enables indicating relative quality rank recommendation, relative priority information, or heatmap signalling that is specific for sphere regions, which are referred to as ERP regions in subclause 7.11. The ERP regions are indicated in the samples of the ERP region timed metadata track with a rectangular grid that is relative to monoscopic ERP that has a full sphere coverage and is aligned with the global coordinate axes.

NOTE 1 An ERP region timed metadata can be associated with any type of omnidirectional video track(s). The associated video track(s) need not use the ERP format. The term ERP region merely refers to a sphere region that corresponds to a rectangle in an ERP picture.

NOTE 2 The indicated grid need not be aligned with any boundaries of sub-picture tracks or any coding structures, such as HEVC tiles.

The ERP region timed metadata indicates one of the following information:

— Heat map information of the ERP regions relative to each other, which indicates the viewing statistics of an ERP region.

— Priority rank information of the ERP regions relative to each other, which is intended as a guidance to OMAF players for selecting tracks to download in case of bandwidth degradation.

— Recommendations on selecting ERP regions based on their quality ranks. This type of metadata indicates the content author's choice which ERP regions are more important to be received with a high quality rank value.

NOTE 3 Methods about how different implementations generate and utilize the signalled ERP region timed metadata information is an implementation detail and outside the scope of this document.

Subclause 8.2.3.2 contains specifications for carriage of ERP region timed metadata in DASH and an OMAF player behaviour description for accessing and using ERP region timed metadata.

7.11.2 Sample entry format

The track sample entry type `'stmd'` shall be used.

The sample entry of this sample type is specified as follows:

```
class ERPRegionTimedMetadataSampleEntry() extends MetaDataSampleEntry ('stmd') {
   unsigned int(8) metadata_type_indicator;
}
```

7.11.3 Semantics

`metadata_type_indicator` indicates the types of timed metadata information which is present in the `ERPRegionTimedMetadataSample` samples that utilize this sample entry.

— (`metadata_type_indicator` & 0x01) equal to 1 indicates a relative quality rank recommendation for the ERP regions.

— (`metadata_type_indicator` & 0x02) equal to 2 indicates a relative priority information for the ERP regions.

— (`metadata_type_indicator` & 0x04) equal to 4 indicates a heatmap signalling for ERP regions, which describes viewing statistics.

NOTE An OMAF player can use heatmaps as relative likelihoods of ERP regions to be viewed by the viewer when selecting which Representations are requested for streaming.

— Value 0 is not allowed.

— (`metadata_type_indicator` & 0x38) shall be equal to 0 in files conforming to this version of this document, and the semantics of (`metadata_type_indicator` & 0x08) equal to 8, (metadata_type_indicator & 0x10) equal to 16, and (metadata_type_indicator & 0x20) equal to 32 are reserved.

— (metadata_type_indicator & 0x40) equal to 64 and (metadata_type_indicator & 0x80) equal to 128 indicate for user-defined metadata types.

7.11.4 Sample format

7.11.4.1 Definition

ERP region timed metadata sample shall be a ERPRegionTimedMetadataSample, which shall be an array of MetadataInformationBoxes.

The types of information which may be present in the sample shall be indicated in the ERPRegionTimedMetadataSampleEntry.

The information of a MetadataInformationBox of a particular metadata_type value is valid up to but excluding the next sample having MetadataInformationBox with the same metadata_type value.

7.11.4.2 Syntax

```
aligned(8) class MetadataInformationBox extends FullBox('mibx', 0, 0) {
   unsigned int(8) metadata_type;
   unsigned int(8) num_rows;
   unsigned int(8) num_cols;
   for (i = 0; i < num_cols; i++)
      for (j = 0; j < num_rows; j++)
         unsigned int(8) value[i][j];
}

aligned(8) class ERPRegionTimedMetadataSample {
   MetadataInformationBox()[];
}
```

7.11.4.3 Semantics

metadata_type specifies the type of the ERP region metadata information. The following values are specified:

— 1 indicates a relative quality rank recommendation for the ERP regions. Higher values indicate that such ERP regions are recommended to be displayed with higher quality rank value compared to the other ERP regions with lower values. A value of 0 indicates that the corresponding ERP region is the least important to be displayed at a high quality rank value.

— 2 indicates a relative priority information for the ERP regions. This value provides a relative priority information of an ERP region compared to the other ERP regions. Higher values indicate that such ERP regions are recommended to be prioritized when it comes to switching to a higher bitrate if there is available bandwidth; and they are recommended to be switched to a lower bitrate later than the other ERP regions with lower priority values if the total bandwidth is degraded. A value of 0 indicates the lowest priority for switching to a higher quality version of the ERP region compared to other ERP regions with higher priority values, and first to switch to a lower quality in case of bandwidth degradation.

— 4 indicates a heatmap signalling for ERP regions. This value is a statistically derived value based on the content viewing characteristics of the viewers who experience the content. The higher the value, the more such an ERP region in the viewport is recommended to be retrieved at higher quality. A value of 0 indicates the lowest heatmap value.

— Values 8, 16 and 32 are reserved.

— Values 64 and 128 indicate user-defined metadata types.

— All values not equal to 1, 2, 4, 8, 16, 32, 64, or 128 are disallowed.

`num_rows` specifies the row partitioning of the coded picture.

`num_cols` specifies the column partitioning of the coded picture.

`value[i][j]` specifies the value assigned to an ERP region which corresponds to the picture partition at index `[i][j]`, where index `[0][0]` corresponds to the top left corner of the coded picture and index `[num_cols - 1][num_rows - 1]` corresponds to the bottom right corner of the coded picture.

7.11.5 Generating ERP region metadata

It is possible to generate ERP region timed metadata in various ways. The following approaches can be given as examples:

a) Viewer analytics: By analysing the content watching pattern of the viewers (e.g. statistics about which ERP regions are in the field-of-view or gaze attention of the viewer at a given time), content creators are able to generate heatmaps, priority ranking or relative quality information.

b) Content analysis: By pre-processing the content with tools such as semantic scene parsing (e.g. detecting and classifying the objects in the scene) or similar computer vision algorithms, content creators can prioritize the visual quality of certain ERP regions (e.g. ERP regions containing people over other ERP regions). Such a prioritization strategy is dependent on the content creators' choice.

Content creators generate and signal ERP region metadata at time intervals of their preference. Each metadata sample corresponds to the information at that given presentation time.

7.12 Storage and signalling of viewpoints for omnidirectional video and images

7.12.1 Viewpoint information structures

7.12.1.1 Definition

The viewpoint information structures provide information of a viewpoint, viewpoint groups, viewpoint switching, and viewpoint looping.

When an edit list is present, it is resolved to prior to considering viewpoint information structures related to timelines.

NOTE The "repeat flag" (`flags` & 1) of the `EditListBox` equal to 1 enables repeating a video track in its entirety. The `ViewpointLoopingStruct()` includes controlling the looping, e.g. with the maximum number of loops and the time period that is looped.

In order to successfully locate the viewpoints, an OMAF player is expected to have access to geolocation tracking and magnetometer. This enables the player to align the common reference coordinate system with the geolocation coordinates and find the player device position with respect to the geolocation coordinates.

7.12.1.2 Viewpoint position structure

The `ViewpointPosStruct()` provides the (X, Y, Z) position of the viewpoint.

```
aligned(8) ViewpointPosStruct() {
   signed int(32) viewpoint_pos_x;
   signed int(32) viewpoint_pos_y;
   signed int(32) viewpoint_pos_z;
}
```

viewpoint_pos_x, viewpoint_pos_y, and viewpoint_pos_z specify the position of the viewpoint in units of 10^{-1} millimetres, in 3D space, relative to the common reference coordinate system. The semantics of this syntax structure depend on the containing syntax structure as follows:

— If ViewpointPosStruct() is contained in ViewpointEntityGroupBox, the viewpoint is static and its position is specified by this ViewpointPosStruct().

— Otherwise, if ViewpointPosStruct() is contained in DynamicViewpointSampleEntry, the viewpoint position may be time-varying and this ViewpointPosStruct() specifies the initial position of the viewpoint.

— Otherwise (when ViewpointPosStruct() is contained in DynamicViewpointSample), this ViewpointPosStruct() specifies the position of the viewpoint for the duration of the sample.

7.12.1.3 Viewpoint GPS position structure

The ViewpointGpsPositionStruct() provides the GPS position of the viewpoint.

```
aligned(8) class ViewpointGpsPositionStruct() {
   signed int(32) viewpoint_gpspos_longitude;
   signed int(32) viewpoint_gpspos_latitude;
   signed int(32) viewpoint_gpspos_altitude;
}
```

viewpoint_gpspos_longitude indicates the longitude of the geolocation of the viewpoint in units of 2^{-23} degrees. viewpoint_gpspos_longitude shall be in range of $-180 * 2^{23}$ to $180 * 2^{23} - 1$, inclusive. Positive values represent eastern longitude and negative values represent western longitude.

viewpoint_gpspos_latitude indicates the latitude of the geolocation of the viewpoint in units of 2^{-23} degrees. viewpoint_gpspos_latitude shall be in range of $-90 * 2^{23}$ to $90 * 2^{23} - 1$, inclusive. Positive value represents northern latitude and negative value represents southern latitude.

viewpoint_gpspos_altitude indicates the altitude of the geolocation of the viewpoint in units of millimetres above the WGS 84 reference ellipsoid.

NOTE The WGS 84 reference ellipsoid is specified in the EPSG:4326 database available at http://www.epsg.org/.

7.12.1.4 Viewpoint geomagnetic information structure

The ViewpointGeomagneticInfoStruct() indicates the orientation of the common reference coordinate system relative to the geomagnetic North direction.

```
aligned(8) class ViewpointGeomagneticInfoStruct() {
   signed int(32) viewpoint_geomagnetic_yaw;
   signed int(32) viewpoint_geomagnetic_pitch;
   signed int(32) viewpoint_geomagnetic_roll;
}
```

viewpoint_geomagnetic_yaw, viewpoint_geomagnetic_pitch, and viewpoint_geomagnetic_roll specify the yaw, pitch, and roll angles, respectively, of the rotation angles of X, Y, Z axes of the common reference coordinate system relative to the geomagnetic North direction, in units of 2^{-16} degrees. viewpoint_geomagnetic_yaw shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive. viewpoint_geomagnetic_pitch shall be in the range of $-90 * 2^{16}$ to $90 * 2^{16}$, inclusive. viewpoint_geomagnetic_roll shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive.

7.12.1.5 Viewpoint global coordinate system rotation structure

The ViewpointGlobalCoordinateSysRotationStruct() provides the yaw, pitch, and roll rotation angles of X, Y, and Z axes, respectively, of the global coordinate system of the viewpoint relative to the common reference coordinate system.

```
aligned(8) class ViewpointGlobalCoordinateSysRotationStruct() {
   signed int(32) viewpoint_gcs_yaw;
   signed int(32) viewpoint_gcs_pitch;
   signed int(32) viewpoint_gcs_roll;
}
```

viewpoint_gcs_yaw, viewpoint_gcs_pitch, and viewpoint_gcs_roll specify the yaw, pitch, and roll angles, respectively, of the rotation angles of X, Y, Z axes of the global coordinate system of the viewpoint relative to the common reference coordinate system, in units of 2^{-16} degrees. viewpoint_gcs_yaw shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive. viewpoint_gcs_pitch shall be in the range of $-90 * 2^{16}$ to $90 * 2^{16}$, inclusive. viewpoint_gcs_roll shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive.

7.12.1.6 Viewpoint group structure

The ViewpointGroupStruct() provides viewpoint group information.

```
aligned(8) class ViewpointGroupStruct(groupDescrIncludedFlag) {
   unsigned int(8) vwpt_group_id;
   if (groupDescrIncludedFlag)
      utf8string vwpt_group_description;
}
```

vwpt_group_id indicates the identifier of a viewpoint group. All viewpoints in a viewpoint group share a common reference coordinate system.

NOTE 1 When two viewpoints have different values of vwpt_group_id, their position coordinates are not comparable, because the viewpoints belong to different coordinate systems.

vwpt_group_description is a null-terminated UTF-8 string which indicates the description of a viewpoint group. A null string is allowed.

NOTE 2 An OMAF player is expected to start with the initial viewpoint timed metadata as defined in subclause 7.12.3.2. Subsequently, if the user wishes to switch to a viewpoint group and the initial viewpoint information is not present, the OMAF player is expected to switch to the viewpoint with the lowest value of the viewpoint identifier in the viewpoint group. Likewise, if an OMAF player starts the playback from a viewpoint group for which no initial viewpoint timed metadata is present, the OMAF player is expected to select the viewpoint with the lowest value of the viewpoint identifier of the viewpoint group.

7.12.1.7 Viewpoint switching list structure

7.12.1.7.1 Definition

The ViewpointSwitchingListStruct(urlIncludedFlag) provides viewpoint switching information.

7.12.1.7.2 Syntax

```
aligned(8) class ViewpointSwitchingListStruct(urlIncludedFlag) {
   unsigned int(8) num_viewpoint_switching;
   for (i = 0; i < num_viewpoint_switching; i++) {
      unsigned int(32) destination_viewpoint_id;
      unsigned int(2) viewing_orientation_in_destination_viewport_mode;
      unsigned int(1) transition_effect_flag;
      unsigned int(1) timeline_switching_offset_flag;
      unsigned int(1) viewpoint_switch_region_flag;
      bit(3) reserved = 0;
      if (viewing_orientation_in_destination_viewport_mode == 1)
         SphereRegionStruct(0,0);
      if (timeline_switching_offset_flag)
         ViewpointTimelineSwitchStruct();
      if (transition_effect_flag) {
         unsigned int(8) transition_effect_type;
         if (transition_effect_type == 4) {
            unsigned int(32) transition_video_track_idx;
            unsigned int(32) transition_audio_track_idx;
         }
         if (transition_effect_type == 5 && urlIncludedFlag)
            utf8string transition_video_URL;
         if (transition_effect_type == 6 && urlIncludedFlag) {
            utf8string transition_scheme_URI;
            utf8string transition_scheme_value;
         }
      }
      if (viewpoint_switch_region_flag) {
         unsigned int(4) num_viewpoint_switch_regions;
         bit(4) reserved = 0;
         for (i = 0; i < num_viewpoint_switch_regions; i++)
            ViewpointSwitchRegionStruct();
      }
   }
}
```

7.12.1.7.3 Semantics

`num_viewpoint_switching` indicates the number of switching transitions possible from the viewpoint to which the `ViewpointSwitchingListStruct()` is associated.

`destination_viewpoint_id` indicates the `viewpoint_id` of the destination viewpoint of a viewpoint switching.

`viewing_orientation_in_destination_viewport_mode` specifies the selection of the viewport after viewpoint switching. The semantics of the values of `viewing_orientation_in_destination_viewport_mode` are specified in Table 14.

Table 14 — Viewing orientation in destination viewport modes

Value	Description
0	Default OMAF viewpoint switching process shall be used (initial viewing orientation if present, or default viewport otherwise).
1	The specific viewing orientation specified by the contained `SphereRegionStruct()` shall be used after transitioning to the new viewpoint.
2	The viewing orientation remains the same in the destination viewpoint as it was in source viewpoint in the common reference coordinate system. This operation is expected to be supported only while switching between viewpoints belonging to the same viewpoint group. This mode is not applicable while switching between viewpoints belonging to different viewpoint groups.
3	Reserved (for use by future extensions of ISO/IEC 23090-2).

`transition_effect_flag` equal to 1 specifies that a transition effect description is present.

`timeline_switching_offset_flag` equal to 1 specifies that a time offset is present.

`viewpoint_switch_region_flag` equal to 1 specifies that a `ViewpointSwitchRegionStruct()` structure is present.

NOTE When the viewpoint switch region is indicated, depending on the client configuration or guided by a user interface, if the user selects the viewpoint switch region, the viewpoint can be switched to the destination viewpoint.

`transition_effect_type` indicates the type of transition effects, as listed in Table 15, when switching to this viewpoint. For `transition_effect_type` values in the range of 0 to 3, inclusive, no semantics or player behaviour is specified, but rather the transitions are described in Table 15.

Table 15 — Viewpoint transition effect types

Value	Description
0	Zoom-in effect: the video first zooms-in the source content, then it is replaced by a zoomed image of the destination content with same zooming factor and finally the destination content is zoomed out.
1	Walk though effect: the video of the destination content replaces the source content by appearing on one side of it (usually in same direction as the movement of the source content) and progressively recovers it entirely by expanding its coverage.
2	Fade-to-black effect: the source content progressively transitions to black (typically by reducing luminosity down to 0), and then the destination content progressively transitions from black to desired luminosity.
3	Mirror effect: the video of the source content is flipped 180 degrees around an axis that runs through the center of the picture (typically a horizontal or vertical axis), and the back face of the flipping plane reveals the destination content.
4	Video transition by playing an indicated video track and, when indicated, also an indicated audio track
5	Video transition by playing the resource indicated by a URL

6	Video transition effect defined by a signalled scheme and value
7..127	Reserved (for use by future extensions of ISO/IEC 23090-2)
128..255	Reserved (for use by external organizations)

`transition_video_track_idx` specifies an index of the track ID contained in the `TrackReferenceTypeBox('vitr')`. The index identifies the video track to be played from time 0 to the end of the track when rendering the transition. The first entry in `TrackReferenceTypeBox('vitr')` has index value equal to 1. `transition_video_track_idx` shall not be equal to 0. The track containing the viewpoint shall include one and only one `TrackReferenceTypeBox('vitr')`.

`transition_audio_track_idx` equal to 0 specifies that no audio is played with the transition video. `transition_audio_track_idx` greater than 0 specifies an index of the track ID contained in the `TrackReferenceTypeBox('autr')`. The index identifies the audio track to be played when rendering the transition. The first entry in `TrackReferenceTypeBox('autr')` has index value equal to 1. When `transition_audio_track_idx` is greater than 0, the track containing the viewpoint shall include one and only one `TrackReferenceTypeBox('autr')`.

`transition_video_URL` indicates the URL of the video to be played when rendering the transition. `transition_video_URL` shall be formatted as an absolute URI as specified in IETF Internet Standard 66. There are no restrictions on the type of the video referred to by the URL.

`transition_scheme_URI` is a null terminated string which indicates the URI which specifies a scheme to interpret the value signalled in `transition_scheme_value`. `transition_scheme_URL` shall be a valid URI as specified in IETF Internet Standard 66. `transition_scheme_URI` shall not be null.

`transition_scheme_value` is a null terminated string which indicates a value which is interpreted based on the scheme signalled in `transition_scheme_URI`. `transition_scheme_value` can be a null string.

`num_viewpoint_switch_regions` specifies the number of switching region `ViewpointSwitchingRegionStruct()` structures for the i-th viewpoint switching. `num_viewpoint_switch_regions` shall not be equal to 0.

7.12.1.7.4 Viewpoint timeline switch structure

The `ViewpointTimelineSwitchStruct()` provides information on how timelines are handled when switching viewpoints.

```
aligned(8) class ViewpointTimelineSwitchStruct() {
   unsigned int(1) relative_t_offset_flag;
   unsigned int(1) min_time_flag;
   unsigned int(1) max_time_flag;
   unsigned int(1) long_time_flag;
   bit(4) reserved=0;
   time_bits = (1 + long_time_flag) * 32;
   if (min_time_flag)
      unsigned int(time_bits) t_min;
   if (max_time_flag)
      unsigned int(time_bits) t_max;
   if (relative_t_offset_flag == 0)
      unsigned int(time_bits) absolute_t_offset;
   else
      int(64) relative_t_offset;
}
```

`relative_t_offset_flag` equal to 0 indicates that the time offset is absolute with respect to the beginning of the stream. When `relative_t_offset_flag` is equal to 1 it indicates that the time offset is relative with respect to the current playout time.

`min_time_flag` equal to 1 specifies that a minimum time with regards to the current viewport media timeline is present. If present, the transition can only be activated after this minimum playout time.

`max_time_flag` equal to 1 specifies that a maximum time with regards to the current viewport media timeline is present. If present, the transition can only be activated before this maximum playout time.

`long_time_flag` equal to 0 specifies that `t_min`, `t_max`, and `absolute_t_offset` are 32-bit unsigned integers. `long_time_flag` equal to 1 specifies that `t_min`, `t_max`, and `absolute_t_offset` are 64-bit unsigned integers.

`t_min` specifies the minimum playout time of the current viewport media timeline after which the switching can be activated.

`t_max` specifies the maximum playout time of the current viewport media timeline before which the switching can be activated.

`absolute_t_offset` specifies the absolute time offset to be used when switching to `destination_viewpoint_id`.

`relative_t_offset` specifies the relative time offset to be used when switching to `destination_viewpoint_id`. The value of `relative_t_offset` is relative with respect to the current playout time. `relative_t_offset` is indicated in the destination viewpoint timescale.

7.12.1.7.5 Viewpoint switch region structure

The `ViewpointSwitchRegionStruct()` indicates a switch region that initiates viewpoint switching when selected by the user. The following region types are specified for a switch region: a region on the viewport, a sphere region, an overlay.

```
aligned(8) class ViewpointSwitchRegionStruct() {
   unsigned int(2)  region_type;
   unsigned int(6)  reserved = 0;
   if (region_type == 0) {
      unsigned int(16) rect_left_percent;
      unsigned int(16) rect_top_percent;
      unsigned int(16) rect_width_percent;
      unsigned int(16) rect_height_percent;
   } else if(region_type == 1) {
      unsigned int(8) shape_type;
      SphereRegionStruct(1,0);
   } else if(region_type == 2)
      unsigned int(16) ref_overlay_id;
}
```

region_type equal to 0 specifies that the viewpoint switch region is positioned on the viewport. region_type equal to 1 specifies that the viewpoint switch region is positioned on the sphere as a sphere region. region_type equal to 2 specifies that the viewpoint switch region is specified by an overlay. Other values of region_type are reserved.

rect_left_percent specifies the coordinates of the left corner of the viewpoint switch region, placed on the viewport in percent relative to the width of the viewport. The values are indicated in units of 100% ÷ 2^{16} in the range of 0 (indicating 0%), inclusive, up to but excluding 65536 (that indicates 100%).

rect_top_percent specifies the coordinates of the top corner of the viewpoint switch region, placed on the viewport in percent relative to the height of the viewport. The values are indicated in units of 100% ÷ 2^{16} in the range of 0 (indicating 0%), inclusive, up to but excluding 65536 (that indicates 100%).

rect_width_percent specifies the width of the viewpoint switch region, placed on the viewport in percent relative to the width of the viewport. The values are indicated in units of 100% ÷ 2^{16} in the range of 1, inclusive, up to but excluding 65536 (that indicates 100%). rect_width_percent shall not be equal to 0.

rect_height_percent specifies the height of the viewpoint switch region, placed on the viewport in percent relative to the height of the viewport. The values are indicated in units of 100% ÷ 2^{16} in the range of 1, inclusive, up to but excluding 65536 (that indicates 100%). rect_height_percent shall not be equal to 0.

(rect_left_percent + rect_width_percent) shall be less than or equal to 65535.

(rect_top_percent + rect_height_percent) shall be less than or equal to 65535.

shape_type is used as the shape type value when applying subclause 7.5.6 to the semantics of the SphereRegionStruct() specifying the viewpoint switch region.

SphereRegionStruct() indicates the sphere region on which the viewpoint switch region is placed. When the SphereRegionStruct() is present, the value of the interpolate syntax element for the SphereRegionStruct() is inferred to be equal to 0.

ref_overlay_id indicates the overlay_id of the overlay that specifies the viewpoint switch region.

7.12.1.8 Viewpoint looping structure

The ViewpointLoopingStruct() provides viewpoint looping information.

```
aligned(8) class ViewpointLoopingStruct (urlIncludedFlag) {
   unsigned int(1) max_loops_flag;
   unsigned int(1) loop_activation_flag;
   unsigned int(1) loop_start_flag;
   unsigned int(1) loop_exit_info_flag;
   bit(4) reserved = 0;
   if (max_loops_flag)
      unsigned int(8) max_loops_num;
   if (loop_activation_flag)
      unsigned int(32) loop_activation_time;
   if (loop_start_flag)
      unsigned int(32) loop_start_time;
   if (loop_exit_info_flag)
      ViewpointSwitchingListStruct(urlIncludedFlag) loop_exit_struct;
}
```

max_loops_flag equal to 1 specifies that information on the maximum number of loops is present.

loop_activation_flag equal to 1 specifies that information on a loop activation time is present.

loop_start_flag equal to 1 specifies that information on a loop start time is present.

loop_exit_info_flag equal to 1 indicates the presence of ViewpointSwitchingListStruct() specifying the viewpoint switching information once the maximum number of loops as defined in max_loops_num is completed. The included ViewpointSwitchingListStruct() shall be constrained to num_viewpoint_switching equal to 1 and viewpoint_switch_region_flag equal to 0.

max_loops_num greater than 0 indicates the maximum number of loops that the OMAF player shall perform on the current viewpoint. When max_loops_num is equal to 0, the maximum number of loops is infinite. When ViewpointLoopingStruct() is present, the OMAF player shall keep looping the current viewpoint until the maximum number of loops is reached or until user interactions with the player or the VR content puts an end to it. When ViewpointLoopingStruct() is present and no max_loops_num is present, the OMAF player shall infer the value of max_loops_num to be equal to 0. It is a constraint of bitstream conformance that when max_loops_num is equal to 0, destination_viewpoint_flag shall be equal to 0.

loop_activation_time indicates the time in the media timeline in units of timescale of the current viewpoint at which the looping shall be initiated. When no loop_activation_time is present, the looping shall occur at the end of the media timeline of the current viewpoint.

NOTE When an OMAF player reaches the inferred or indicated loop activation time the first time, the loop counter is set equal to 1 and the looping is initiated. In each subsequent time when the OMAF player reaches the inferred or indicated loop activation time, the loop counter is incremented by 1. If the loop counter is equal to max_loops_num, the looping is terminated.

loop_start_time indicates the time in the media timeline in units of timescale of the current viewpoint at which playout of the current viewpoint shall restart when a loop is performed. When no loop_start_time is present, playout shall start at the beginning of the media timeline of the current viewpoint.

`loop_exit_struct` indicates the destination viewpoint, the timeline handling, the transition effect and the destination viewport orientation mode of a viewpoint switching that shall occur at the end of the playout of the current viewpoint once the maximum number of loops (`max_loops_num`) has been reached. When no `loop_exit_struct` is present (i.e. when `loop_exit_info_flag` is equal to 0) and the maximum number of loops is reached, the presentation of this viewpoint ends and the following applies:

— If the syntax structure containing this `ViewpointLoopingStruct()` also contains `ViewpointSwitchingListStruct()`, an OMAF player shall display the paused visual content at the end of the loop period until user interactions with the player or the VR content cause switching to the next viewpoint as specified in the `ViewpointSwitchingListStruct()`.

— Otherwise, the presentation of the OMAF content ends.

7.12.2 Viewpoint entity grouping

7.12.2.1 Definition

Tracks and image items that are present in the same entity group with `grouping_type` equal to `'vipo'` belong to the same viewpoint.

When no `'vipo'` entity group is present in a file, the file contains content for one viewpoint only.

When tracks or image items belonging to more than one viewpoint are present in a file, a `'vipo'` entity group shall be present for each viewpoint. A `'vipo'` entity group for a particular viewpoint shall list all tracks and image items containing background visual media of the viewpoint and all non-video media tracks of the viewpoint. When a `'dyvp'` timed metadata track is present for a viewpoint, the `'vipo'` entity group for the viewpoint shall include the `'dyvp'` timed metadata track. All video tracks included in a `'vipo'` entity group shall have the same timescale, referred to as the viewpoint timescale.

When both a `'vipo'` entity group for a particular viewpoint and an ERP region timed metadata track applying to that viewpoint are present, the `'vipo'` entity group shall include the ERP region timed metadata track. When no `'vipo'` entity group is present, an ERP region timed metadata track, if any, applies to the only viewpoint that is present in the file.

7.12.2.2 Syntax

```
aligned(8) class ViewpointEntityGroupBox extends EntityToGroupBox('vipo',0,0) {
   unsigned int(32) viewpoint_id;
   utf8string viewpoint_label;
   ViewpointInformationStruct(1,1);
}

aligned(8) ViewpointInformationStruct(groupDescrIncludedFlag,
urlIncludedFlag)
{

   unsigned int(1) gpspos_present_flag;
   unsigned int(1) geomagnetic_info_present_flag;
   unsigned int(1) switching_info_present_flag;
   unsigned int(1) looping_present_flag;
   bit(4) reserved = 0;

   ViewpointPosStruct();

   ViewpointGroupStruct(groupDescrIncludedFlag);

   ViewpointGlobalCoordinateSysRotationStruct();
```

```
    if(gpspos_present_flag)
        ViewpointGpsPositionStruct();

    if(geomagnetic_info_present_flag)
        ViewpointGeomagneticInfoStruct();

    if(switching_info_present_flag)
        ViewpointSwitchingListStruct(urlIncludedFlag);

    if(looping_present_flag)
        ViewpointLoopingStruct();
}
```

7.12.2.3 Semantics

`viewpoint_id` specifies the viewpoint identifier of the viewpoint. The value of `viewpoint_id` shall differ from the values of `viewpoint_id` in all other the `'vipo'` entity groups.

`viewpoint_label` is a null-terminated UTF-8 string that provides a human readable text label for the viewpoint.

`gpspos_present_flag` equal to 1 indicates that `ViewpointGpsPositionStruct()` is present. `gpspos_present_flag` equal to 0 indicates that that `ViewpointGpsPositionStruct()` is not present.

`geomagnetic_info_present_flag` equal to 1 indicates that `ViewpointGeomagneticInfoStruct()` is present. `geomagnetic_info_present_flag` equal to 0 indicates that `ViewpointGeomagneticInfoStruct()` is not present. `geomagnetic_info_present_flag` should be equal to 1 for at most one viewpoint in viewpoint group to enable location aware VR content.

`global_coord_rotation_present_flag` equal to 1 indicates that `ViewpointGlobalCoordinateSysRotationStruct()` is present. `global_coord_rotation_present_flag` equal to 0 indicates that that `ViewpointGlobalCoordinateSysRotationStruct()` is not present.

`switching_info_present_flag` equal to 1 indicates that `ViewpointSwitchingListStruct()` is present. `switching_info_present_flag` equal to 0 indicates that `ViewpointSwitchingListStruct()` is not present.

When `ViewpointSwitchingListStruct()` is not present it means that there is no recommended behaviour when switching between viewpoints and an OMAF player can freely switch between this viewpoint and any other viewpoint as desired.

`looping_present_flag` equal to 1 indicates that `ViewpointLoopingStruct()` is present. `looping_present_flag` equal to 0 indicates that `ViewpointLoopingStruct()` is not present. When all entities in a `'vipo'` entity group are image items, `looping_present_flag` shall be equal to 0.

When `ViewpointLoopingStruct()` is not present, no looping behaviour for the viewpoint is specified through the `ViewpointEntityGroup` structure.

NOTE When a track contains an edit list and the "repeat flag" (`flags` & 1) of the `EditListBox` is equal to 1, the edit list repeated until the track duration is reached. When `ViewpointLoopingStruct()` is not present, an OMAF player is expected to play the viewpoint, as modified by the edit list (if any), once till end of the movie timeline.

7.12.3 Timed metadata for viewpoints

7.12.3.1 Dynamic viewpoint information

7.12.3.1.1 General

The dynamic viewpoint timed metadata track indicates the viewpoint parameters that are dynamically changing over time.

7.12.3.1.2 Sample entry

The track sample entry type 'dyvp' shall be used. The sample entry of this sample entry type is specified as follows:

```
class DynamicViewpointSampleEntry extends MetaDataSampleEntry('dyvp') {
   unsigned int(1) vwpt_pos_flag;
   unsigned int(1) dynamic_gcs_rotation_flag;
   unsigned int(1) dynamic_vwpt_group_flag;
   unsigned int(1) dynamic_vwpt_gps_flag;
   unsigned int(1) dynamic_vwpt_geomagnetic_info_flag;
   unsigned int(1) dynamic_vwpt_switch_flag;
   bit(2) reserved = 0;
   if (vwpt_pos_flag)
      ViewpointPosStruct();
   if (!dynamic_gcs_rotation_flag)
      ViewpointGlobalCoordinateSysRotationStruct();
   if (!dynamic_vwpt_gps_flag)
      ViewpointGpsPositionStruct();
   if (!dynamic_vwpt_geomagnetic_info_flag)
      ViewpointGeomagneticInfoStruct();
   if (!dynamic_vwpt_switch_flag)
      ViewpointSwitchingListStruct(1);
}
```

vwpt_pos_flag equal to 0 specifies that ViewpointPosStruct() is not present in the sample entry. vwpt_pos_flag equal to 1 specifies that ViewpointPosStruct() is present in the sample entry.

dynamic_gcs_rotated_flag equal to 0 specifies that the yaw, pitch, and roll rotation angles of X, Y, and Z axes, respectively, of the global coordinate system of the viewpoint relative to the common reference coordinate system remain unchanged in all samples referring to this sample entry. dynamic_gcs_rotated_flag equal to 1 specifies that the yaw, pitch, and roll rotation angles of X, Y, and Z axes, respectively, of the global coordinate system of the viewpoint relative to the common reference coordinate system are indicated in the samples.

dynamic_vwpt_group_flag equal to 0 specifies that the vwpt_group_flag and ViewpointGroupStruct() are not present in the samples and the viewpoint group information (vwpt_group_id and vwpt_group_description) signalled in ViewpointGroupStruct() in the ViewpointEntityGroupBox applies to each sample referring to this sample entry. dynamic_vwpt_group_flag equal to 1 specifies that the ViewpointGroupStruct() for the viewpoint may change compared to that signalled in the ViewpointGroupStruct() in the ViewpointEntityGroupBox and new ViewpointGroupStruct() may be signalled in the samples based on the value of vwpt_group_flag in the samples.

dynamic_vwpt_gps_flag equal to 0 specifies that the GPS position information of the viewpoint remains unchanged or is not updated in all samples referring to this sample entry. dynamic_vwpt_gps_flag equal to 1 specifies that the GPS position information of the viewpoint is indicated in the samples.

dynamic_vwpt_geomagnetic_info_flag equal to 0 specifies that ViewpointGeomagneticInfoStruct() is present in the sample entry.

dynamic_vwpt_geomagnetic_info_flag equal to 1 specifies that ViewpointGeomagneticInfoStruct() is not present in the sample entry and is indicated in the samples.

dynamic_vwpt_switch_flag equal to 0 specifies that ViewpointSwitchingListStruct() is present in the sample entry and is not present in the samples. dynamic_vwpt_switch_flag equal to 1 specifies that ViewpointSwitchingListStruct() is not present in the sample entry and is present in the samples.

ViewpointPosStruct() is specified in subclause 7.12.1.2 and indicates the initial viewpoint position.

ViewpointGlobalCoordinateSysRotationStruct() is defined in subclause 7.12.1.5 but indicates the yaw, pitch, and roll rotation angles of X, Y, and Z axes, respectively, of the global coordinate system of the viewpoint relative to the common reference coordinate system for each sample referring to this sample entry.

ViewpointGpsPositionStruct() is defined in subclause 7.12.1.3 but indicates the GPS position information of the viewpoint for each sample referring to this sample entry.

ViewpointGeomagneticInfoStruct() is defined in subclause 7.12.1.4 but indicates the geomagnetic position information of the viewpoint for each sample referring to this sample entry.

ViewpointSwitchingListStruct() is defined in subclause 7.12.1.7 but indicates the viewpoint switching information for each sample referring to this sample entry.

7.12.3.1.3 Sample format

The sample syntax of this sample entry type ('dyvp') is specified as follows:

```
aligned(8) DynamicViewpointSample() {
   ViewpointPosStruct();
   if (dynamic_vwpt_gps_flag)
      ViewpointGpsPositionStruct();
   if (dynamic_vwpt_geomagnetic_info_flag)
      ViewpointGeomagneticInfoStruct();
   if (dynamic_gcs_rotation_flag)
      ViewpointGlobalCoordinateSysRotationStruct();
   if (dynamic_vwpt_switch_flag)
      ViewpointSwitchingListStruct(1);
   if (dynamic_vwpt_group_flag) {
      unsigned int(1) vwpt_group_flag;
      bit(7) reserved = 0;
      if (vwpt_group_flag)
         ViewpointGroupStruct(1);
   }
}
```

The semantics of ViewpointPosStruct(), ViewpointGpsPositionStruct(), ViewpointGeomagneticInfoStruct(), ViewpointGlobalCoordinateSysRotationStruct(), ViewpointSwitchingListStruct(), and ViewpointGroupStruct() are specified in subclause 7.12.1 and apply for the duration of the sample.

vwpt_group_flag equal to 1 specifies that the ViewpointGroupStruct() is present. vwpt_group_flag equal to 0 specifies that the ViewpointGroupStruct() is not present. When not present, the value of vwpt_group_flag is inferred to be equal to 0.

When dynamic_vwpt_group_flag is equal to 1, the first sample shall have vwpt_group_flag equal to 1. For subsequent samples when the viewpoint group information does not change, the ViewpointGroupStruct() can be absent. When the ViewpointGroupStruct() is absent in a sample, the following applies:

— If dynamic_vwpt_group_flag is equal to 1, it is inferred to be identical to the ViewpointGroupStruct() of the previous sample, in decoding order and the value of groupDescrIncludedFlag is inferred to be identical to the value of the groupDescrIncludedFlag of the ViewpointGroupStruct() of the previous sample .

— Otherwise (dynamic_vwpt_group_flag is equal to 0), it is inferred to be identical to the ViewpointGroupStruct() in the ViewpointEntityGroupBox and the value of groupDescrIncludedFlag is inferred to be identical to the value of the groupDescrIncludedFlag of the ViewpointGroupStruct() in the ViewpointEntityGroupBox.

7.12.3.2 Initial viewpoint

7.12.3.2.1 General

This timed metadata track, named the initial viewpoint timed metadata track, indicates the initial viewpoint that should be used.

When viewpoints are not indicated by the viewpoint entity grouping in a file, the initial viewpoint timed metadata track shall not be present in the file.

In the absence of this information when viewpoints are indicated by the viewpoint entity grouping, the initial viewpoint should be inferred to be the viewpoint that has the least value of viewpoint identifier among all viewpoints in the file.

The initial viewpoint timed metadata track, when present, is associated with all viewpoints in the file.

7.12.3.2.2 Sample entry

The track sample entry type 'invp' shall be used. The sample entry of this sample entry type is specified as follows:

```
class InitialViewpointSampleEntry extends MetaDataSampleEntry('invp') {
   unsigned int(32) id_of_initial_viewpoint;
}
```

id_of_initial_viewpoint indicates the value of the viewpoint identifier of the initial viewpoint for the first sample to which this sample entry applies.

7.12.3.2.3 Sample format

The sample syntax of this sample entry type ('invp') is specified as follows:

```
aligned(8) InitialViewpointSample() {
   unsigned int(32) id_of_initial_viewpoint;
}
```

id_of_initial_viewpoint indicates the value of the viewpoint identifier of the initial viewpoint for the sample.

7.12.3.3 Object centre points correspondence between viewpoints

7.12.3.3.1 General

This timed metadata track, named the object centre points correspondence (OCPC) timed metadata track, indicates information on object centre points correspondence between viewpoints. An OCPC timed metadata track applies to all omnidirectional video tracks in the file. The OCPC timed metadata track describes the object centre points correspondence for important objects of interest. This enables to maintain the switching continuity between different viewpoints by maintaining the viewport oriented towards the selected object(s) of interest.

7.12.3.3.2 Sample entry

The track sample entry type `'ocpc'` shall be used. The sample entry of this sample entry type is specified as follows:

```
class OcpcSampleEntry() extends SphereRegionSampleEntry('ocpc') {
   unsigned int(31) num_viewpoint_sets;
   unsigned int(1) last_byte_sent;
   unsigned int(8) num_object_ids;
   for (m = 0; m < num_object_ids; m++) {
      unsigned int(8) object_id[m];
      utf8string object_label[m];
   }
}
```

`num_viewpoint_sets` indicates the number of viewpoint sets for which object centre points correspondence is signalled in the samples to which this sample entry applies.

`last_byte_sent` equal to 1 specifies that the last byte of the sphere region structure, which includes the `interpolate` bit and 7 reserved bits, is signalled in `OcpcSample()`'s sphere region structure. `last_byte_sent` equal to 0 specifies that the last byte of the sphere region structure, which includes the `interpolate` bit and 7 reserved bits, is not signalled in `OcpcSample()`'s sphere region structure.

`num_object_ids` specifies the number of object identifiers for which `object_id` and `object_label` values are signalled.

`object_id[m]` specifies the object identifier for the m-th object. The value of `object_id[i][k][j]` for each `i` in the range of 0 to `num_viewpoint_sets − 1`, inclusive, for each `j` in the range of 0 to `num_viewpoints_in_this_set[i] − 1`, inclusive, and for each `k` in the range of 0 to `num_corresp_object_centre_points_in_this_set[i] − 1`, inclusive, shall match one value of `object_id[m]` for `m` in the range of 0 to `num_object_ids − 1`, inclusive.

`object_label[m]` is a null-terminated UTF-8 string that specifies the description of an object identified by `object_id[m]`. A null string is allowed.

7.12.3.3.3 Sample format

The sample syntax of this sample entry type (`'ocpc'`) is specified as follows:

```
class OcpcSample() {
   for (i = 0; i < num_viewpoint_sets; i++) {
      unsigned int(8) num_viewpoints_in_this_set[i];
      for (j = 0; j < num_viewpoints_in_this_set[i]; j++)
         unsigned int(32) viewpoint_id[i][j];
      unsigned int(16) num_corresp_object_centre_points_in_this_set[i];
      for (k = 0; k < num_corresp_object_centre_points_in_this_set[i]; k++)
         for (j = 0; j < num_viewpoints_in_this_set[i]; j++) {
            SphereRegionStruct(0, last_byte_sent)[i][k][j];
            unsigned int(8) object_id[i][k][j];
         }
   }
}
```

`num_viewpoints_in_this_set[i]` indicates the number of viewpoints in the i-th viewpoint set.

`viewpoint_id[i][j]` indicates the viewpoint ID of the j-th viewpoint in the i-th viewpoint set.

`num_corresp_object_centre_points_in_this_set[i]` indicates the number of corresponding object centre points signalled in this sample for the i-th viewpoint set.

SphereRegionStruct(0, last_byte_sent)[i][k][j] specifies the k-th corresponding object centre point of the j-th viewpoint in the i-th viewpoint set.

object_id[i][k][j] specifies the k-th object identifier of the j-th viewpoint in the i-th viewpoint set.

interpolate equal to 0 specifies that the values of centre_azimuth, centre_elevation, and centre_tilt in the OcpcSample specify the object centre at the sample instance. interpolate equal to 1 specifies that the values of centre_azimuth, centre_elevation, and centre_tilt in the OcpcSample can be linearly interpolated from the values of the corresponding fields in this sample and the previous sample for estimating the information about object centres in between those two OcpcSample()s.

When last_byte_sent is equal to 0, interpolate is inferred to be equal to 0.

For any particular value of k in the range of 0 to num_corresp_object_centre_points_in_this_set[i] − 1, inclusive, the sphere points indicated by SphereRegionStruct(0, last_byte_sent)[i][k][j] for j ranging from 0 to num_viewpoints_in_this_set[i] − 1, inclusive, are object centre points that are corresponding to each other for the viewpoints in the i-th viewpoint set.

7.12.3.3.4 Information derivation and OMAF player behaviour

Content providers can perform scene or object matching among video streams representing different viewpoints frame by frame, and choose a representative point of the scene or object, e.g. the centre point of an object, as the corresponding object centre point to be indicated by the OCPC timed metadata track. The object centre point may be assigned an object label to indicate what the object represents.

The available object labels (i.e. object_label[m]) or object identifiers (i.e. object_id[m]) for each viewpoint may be shown to the user via a user interface. The user should be allowed to select an object of interest to view by selecting shown object label or object identifier. The viewport corresponding to the selected object label or object identifier’s object centre should be shown to the user for the selected viewpoint. When a viewpoint switching occurs, the client checks whether the user's field of view in the switch-from viewpoint covers a corresponding object centre point that is indicated by the time-aligned sample of the OCPC timed metadata track. If yes, just after the switching, the client should render to the user the viewport in the switching-to viewpoint for which the corresponding centre point is indicated by the time-aligned sample of the OCPC timed metadata track. When the user's field of view covers more than one indicated object centre point, one of those that are the closest to the centre of the user's field of view should be chosen.

If both recommended viewport metadata information for the switch-to viewpoint and the OCPC timed metadata track are available, since both types of information do not impose mandatory OMAF player behaviour, then it is up to the OMAF player to choose to follow either one or no one.

7.13 Storage of omnidirectional video in sub-picture tracks

7.13.1 General

For storage of omnidirectional video in sub-picture tracks, a SpatialRelationship2DDescriptionBox with track_group_type equal to '2dsr' as specified in subclause 7.1.4.2 shall be present in each sub-picture track, to indicate that this track belongs to a group of tracks that can be spatially arranged to construct composition pictures. For each sub-picture track storing a part of an omnidirectional video, the SubPictureRegionBox shall be present in SpatialRelationship2DDescriptionBox and the sub-picture track shall not contain an instance of SpatialRelationship2DGroupEntry or SampleToGroupBox with grouping_type equal to '2dsr' as specified in subclause 7.1.4.3.

7.13.2 Projected omnidirectional video

This subclause applies when any of the tracks mapped to the 2D spatial relationship track group has a sample entry type equal to 'resv' and scheme_type equal to 'podv' in the SchemeTypeBox included in the sample entry.

The projection format of each composition picture is indicated by any ProjectionFormatBox.

The region-wise packing format is indicated by the collection of the instances of RegionWisePackingBox within the sample entries of the sub-picture tracks mapped to the same 2D spatial relationship track group, and the region-wise packing information for the composition picture is inferred for each region i in the range of 0 to the number of sub-picture tracks in the '2dsr' track group, exclusive, in any order as follows:

— packed_picture_width and packed_picture_height for the composition picture are inferred to be equal to total_width and total_height of SpatialRelationship2DSourceBox, respectively.

— packed_reg_top[i] and packed_reg_left[i] are inferred to be respectively equal to object_y and object_x of SubPictureRegionBox of SpatialRelationship2DDescriptionBox of the respective sub-picture track.

— The other syntax elements of each region of the composition picture are inferred to be equal to the respective syntax elements of the RegionWisePackingBox of the respective sub-picture track.

When applicable, the frame packing arrangement of each composition picture is indicated by any StereoVideoBox within the sample entry of any track of the same 2D spatial relationship track group.

The values of object_width and object_height of SubPictureRegionBox in SpatialRelationship2DDescriptionBox shall be equal to the width and height, respectively, of the pictures output by the decoder in luma sample units.

The following constraints apply for the sub-picture tracks mapped to the same 2D spatial relationship track group:

— Each track mapped to this grouping shall have a sample entry type equal to 'resv'. The scheme_type shall be equal to 'podv' in the SchemeTypeBox included in the sample entry.

— The content of all instances of the ProjectionFormatBox included in the sample entries of the tracks mapped to the same 2D spatial relationship track group shall be identical.

— RegionWisePackingBox shall be present in the sample entries of the tracks mapped to a '2dsr' track group. The values of proj_picture_width shall be identical in all these instances of RegionWisePackingBox. The values of proj_picture_height shall be identical in all these instances of RegionWisePackingBox.

— The content of all instances of the RotationBox included in the sample entries of the tracks mapped to the same 2D spatial relationship track group shall be identical.

— The content of all instances of the StereoVideoBox included in the sample entries of the tracks mapped to the same 2D spatial relationship track group shall be identical.

— The content of all instances of the CoverageInformationBox included in all instances of the SpatialRelationship2DDescriptionBox in the tracks mapped to the same 2D spatial relationship track group shall be identical.

— When one track mapped to a 2D spatial relationship track group has a CompositionInformationBox included in SpatialRelationship2DDescriptionBox, all tracks mapped to the same 2D spatial relationship track group shall have a CompositionInformationBox included in SpatialRelationship2DDescriptionBox.

7.13.3 Indication of composition pictures being packed pictures or projected pictures

The `CompositionInformationBox` may be optionally included in `SpatialRelationship2DDescriptionBox`. The presence of `CompositionInformationBox` in `SpatialRelationship2DDescriptionBox` indicates that the composition picture is a projected picture. The absence of `CompositionInformationBox` in `SpatialRelationship2DDescriptionBox` indicates that the composition picture is a packed picture.

NOTE 1 `CoverageInformationBox` included in `SpatialRelationship2DDescriptionBox` can be used for indicating content coverage when the composition picture does not cover the entire sphere.

NOTE 2 If `CompositionInformationBox` is present, OMAF players can reconstruct projected pictures without parsing the `RegionWisePackingBox`es of the associated sub-picture tracks. Otherwise, OMAF players have to parse `RegionWisePackingBox`es of the associated sub-picture tracks to reconstruct projected pictures.

The syntax of `CompositionInformationBox` is as follows:

```
aligned(8) class CompositionInformationBox extends FullBox('coif', 0, 0) {
}
```

7.13.4 Fisheye omnidirectional video

This subclause applies when any of the tracks mapped to the 2D spatial relationship track group has a sample entry type equal to `'resv'` and `scheme_type` equal to `'fodv'` in the `SchemeTypeBox` included in the sample entry.

The following constraints apply for the tracks mapped to this grouping:

— Each track mapped to this grouping shall have a sample entry type equal to `'resv'`. The `scheme_type` shall be equal to `'fodv'` in the `SchemeTypeBox` included in the sample entry.

— All tracks mapped to the 2D spatial relationship track group shall have a `FisheyeOmniVideoBox` included in the sample entry.

— A `FisheyeOmniVideoBox` that specifies the fisheye format of the composition picture may be contained in `SpatialRelationship2DDescriptionBox`. When one track mapped to a 2D spatial relationship track group has a `FisheyeOmniVideoBox` included in `SpatialRelationship2DDescriptionBox`, all tracks mapped to the same 2D spatial relationship track group shall have a `FisheyeOmniVideoBox` included in `SpatialRelationship2DDescriptionBox` and the content of all these instances of the `FisheyeOmniVideoBox` shall be identical.

  NOTE A `FisheyeOmniVideoBox` included in the sample entry describes the fisheye format of the track.

— The `SpatialRelationship2DDescriptionBox` shall not include `RegionWisePackingBox`, `CoverageInformationBox`, or `CompositionInformationBox`.

7.14 Storage and signalling of overlays for omnidirectional video and images

7.14.1 General

Subclause 7.14 specifies support for overlay video and overlay images, which are superimposed either over omnidirectional background video or image or on the viewport.

An overlay is specified with the following pieces of information:

— overlay source, specifying which track or image item and, in some cases, which spatial region within the track or the image item are used as the content of the overlay

— rendering type of the overlay, specifying whether the overlay is anchored relative to the viewport or to the unit sphere and, for sphere-relative overlays, the shape of the surface on which the overlay is rendered (plane or sphere region)

— rendering properties of the overlay, specifying e.g. the opacity of the overlay

— user interaction properties for the overlay

The overlay source is one of the following:

— entire decoded pictures of a video track

— an entire output image of an image item

— an indicated rectangular overlay source region within decoded pictures of a video track or an output image of an image item

— a recommended viewport of omnidirectional video, as indicated by a recommended viewport timed metadata track

— an externally specified overlay source, referenced by a URI

When a source region is used as overlay source, the picture that contains the source region may also contain other overlay source regions or background visual media.

The rendering type of the overlay is one of the following:

— viewport-relative overlay, specifying that the overlay is displayed on a rectangular area at a specified position relative to the viewport

— sphere-relative projected omnidirectional overlay, specifying that the overlay is displayed on a sphere surface at an indicated position within or on the unit sphere

— sphere-relative 2D overlay, specifying that the overlay is displayed on a plane at an indicated position within the unit sphere

— 3D mesh overlay, specifying that the overlay is displayed on indicated mesh elements of the 3D mesh

The following rendering properties may be indicated for an overlay:

— layering order, specifying the layering order among the overlays that are relative to the viewport, and separately among each set of overlays that have the same distance from the centre of the unit sphere

— opacity of the overlay that is applied for all pixels of the overlay source when rendering the overlay

— priority value (`overlay_priority`), based on which a priority order of overlays can be derived and used in the case that an OMAF player does not have enough decoding capacity to decode all overlays; value 0 indicates that the overlay is essential for displaying

— alpha composition, specifying that the overlay is associated with an alpha plane used for determining pixel-wise opacity and blending when superimposing the overlay

The following user interaction properties may be indicated for an overlay:

— controls which user interactions are allowed in a user interface

— overlay label, which can be used in a user interface

— associated sphere region, intended to be used as a user-selectable area to turn an overlay on or off

The overlay source, rendering type, rendering properties and user interaction properties for an overlay are specified as overlay controls in the `SingleOverlayStruct()` syntax structure. `SingleOverlayStruct()` is extensible, i.e. new overlay controls can be introduced in future version of this document. Overlay controls are indicated to be non-essential or essential. In latter case, an OMAF player is required to process the overlay control. A content author can specify an overlay control to be non-essential e.g. when applying the overlay control for rendering is preferred but not mandated. `SingleOverlayStruct()` additionally contains an overlay identifier (`overlay_id`). Since the same video track or image item may contain multiple overlays, the `OverlayStruct()` syntax structure specifies the number of overlays and includes a `SingleOverlayStruct()` for each of them.

Overlay controls can be static or time-varying. Static overlay controls are contained in a sample entry in a track containing the overlay source or as an item property for an image item containing the overlay source. Time-varying overlay controls are included in a timed metadata track.

The tracks and image items containing associated overlays and the background visual media are indicated in an `'ovbg'` entity group specified in subclause 7.14.7.2.

An overlay can be either active or inactive, which is determined as follows:

— If both of the following conditions are true, an overlay is active for a particular background visual media:

  — The overlay is included among the overlays associated with the background visual media in its `'ovbg'` entity group.

  — There is no overlay timed metadata track for the overlay or the overlay is activated by an overlay timed metadata track.

  NOTE As an alternative of using a timed metadata track to indicate the time period when an overlay image item is active, the active time period associated with the overlay image item can be indicated through a video track containing a sample that encloses the coded data of the image item.

— Otherwise, the overlay is inactive for the background visual media.

An inactive overlay shall not be displayed. An OMAF player shall display an active overlay when either of the following conditions is true at any particular time:

— `overlay_priority` is equal to 0 for the overlay and the overlay is not turned off by the user through a user interface.

— `overlay_priority` is greater than 0, an OMAF player has enough capacity for decoding and rendering the overlay, and the overlay is not turned off by the user through a user interface.

In addition to the active/inactive status for overlays, an overlay has a state on or off, which indicates whether an overlay has been turned on or off by a user interaction. An overlay is turned on by default. The overlay controls for user interactions include a control that enables or disables the user action for switching an overlay on or off. When an overlay is turned off by a user interaction, it is not displayed.

Subclause 7.14 specifies the syntax and semantics of overlay-related syntax structures. Subclause 7.15 specifies a viewing space intended to be used for head-tracked rendering of background visual media and overlays. Annex G specifies a rendering procedure for overlays.

7.14.2 Overlay structure

7.14.2.1 Definition

`OverlayStruct()` specifies the overlay related metadata per each overlay.

7.14.2.2 Syntax

```
aligned(8) class SingleOverlayStruct() {
   unsigned int(16) overlay_id;
   for (i = 0; i < num_flag_bytes * 8; i++)
      unsigned int(1) overlay_control_flag[i];
   for (i = 0; i < num_flag_bytes * 8; i++){
      if (overlay_control_flag[i]) {
         unsigned int(1) overlay_control_essential_flag[i];
         unsigned int(15) byte_count[i];
         unsigned int(8) overlay_control_struct[i][byte_count[i]];
      }
   }
}

aligned(8) class OverlayStruct() {
   unsigned int(16) num_overlays;
   unsigned int(8) num_flag_bytes;
   for (i = 0; i < num_overlays; i++)
      SingleOverlayStruct();
}
```

7.14.2.3 Semantics

`num_overlays` specifies the number of overlays described by this structure. `num_overlays` equal to 0 is reserved.

`num_flag_bytes` specifies the number of bytes allocated collectively by the `overlay_control_flag[i]` syntax elements. OMAF players shall parse the syntax of `OverlayStruct()` regardless of the value of `num_flag_bytes`.

NOTE 1 In files conforming to this version of this document, `num_flag_bytes` is expected to be equal to 1 or 2, depending on which overlay control structures are present.

`overlay_id` provides a unique identifier for the overlay. No two overlays shall have the same `overlay_id` value.

`overlay_control_flag[i]` when set to 1 defines that the structure as defined by the i-th `overlay_control_struct[i]` is present. This version of this document specifies the semantics of overlay controls in the range of 0 to OVLY_LAST_ED2_IDX, inclusive, where OVLY_LAST_ED2_IDX is equal to 14. In files conforming to this version of this document, `overlay_control_flag[i]` shall not be equal to 1 when `i` is greater than OVLY_LAST_ED2_IDX. OMAF players shall allow both values of `overlay_control_flag[i]` to appear in the syntax for all values of `i`.

NOTE 2 When the value of `i` is greater than OVLY_LAST_ED2_IDX and `overlay_control_essential_flag[i]` is equal to 0, an OMAF player implementing this version of this document needs not process `overlay_control_flag[i]` and `overlay_control_struct[i]`. When the value of `i` is greater than OVLY_LAST_ED2_IDX and `overlay_control_essential_flag[i]` is equal to 1, the OMAF player requirements specified below in this subclause apply.

NOTE 3 Since OMAF players conforming to the this version of this document allow `num_flag_bytes` greater than 2 and `overlay_control_flag[i]` equal to 1 for all values of i to appear in the syntax, OMAF players conforming to this version of this document:

– are able to parse OverlayStruct() conforming to potential future OMAF editions or amendments, where num_flag_bytes can be greater than 2 or overlay_control_flag[i] can be equal to 1 for values of i greater than OVLY_LAST_ED2_IDX; and

– are permitted to display an overlay indicated in the OverlayStruct(), when overlay_control_essential_flag[i] is not equal to 1 for any value of i greater than OVLY_LAST_ED2_IDX.

overlay_control_essential_flag[i] equal to 0 specifies that OMAF players are not required to process the structure as defined by the i-th overlay_control_struct[i]. overlay_control_essential_flag[i] equal to 1 specifies that OMAF players shall process the structure as defined by the i-th overlay_control_struct[i].

When overlay_control_flag[i] with i in the range of 0 to 6, inclusive, or equal to 11 is equal to 1, the value of overlay_control_essential_flag[i] shall be equal to 1.

OMAF players shall be able to parse and process the structure corresponding to overlay_control_struct[11] (i.e. the OverlayPriority control structure).

When overlay_control_essential_flag[i] is equal to 1 and an OMAF player is not capable of parsing or processing the structure as defined by overlay_control_struct[i], the following applies:

— If the OverlayPriority control structure is present for the overlay and overlay_priority in OverlayPriority is equal to 0, the OMAF player shall display neither the overlays specified by this OverlayStruct() nor the background visual media.

— Otherwise, the OMAF player shall not display the overlay specified by this SingleOverlayStruct().

byte_count[i] gives the byte count of the structure represented by the i-th overlay_control_struct[i]. byte_count[4], when present, shall be equal 0.

overlay_control_struct[i][byte_count[i]] defines the i-th structure with a byte count as defined by byte_count[i].

7.14.3 Overlay control structures

7.14.3.1 Definition

Table 16 specifies the semantics of bit index i, for each value of i in the range of 0 to 14, inclusive. When overlay_control_flag[i] is equal to 1, the overlay control as specified in Table 16 applies. Furthermore, Table 16 specifies the subclause that applies for the syntax and semantics of the overlay control structure for each bit index.

Table 16 — Overlay control structures

Bit index	Subclause in this document	Description
0	7.14.3.2	Parameters for viewport-relative overlay
1	7.14.3.3	Parameters for sphere-relative projected omnidirectional overlay
2	7.14.3.4	Parameters for sphere-relative 2D overlay
3	7.14.3.5	Parameters for 3D mesh overlay
4	none	Entire decoded picture is an overlay source.
5	7.14.3.6	Source region for the overlay. Indicates the region within the decoded picture that is used as the content of the overlay.
6	none	Recommended viewport overlay. Indicates the recommended viewport track whose recommended viewports are used as the content of the overlay.
7	7.14.3.7	Overlay layering order
8	7.14.3.8	Overlay opacity
9	7.14.3.9	Controls for user interaction
10	7.14.3.10	Overlay label
11	7.14.3.11	Overlay priority
12	7.14.3.12	Associated sphere region
13	7.14.3.13	Overlay alpha composition
14	7.14.3.14	Externally specified overlay information

The following constraints apply for each `SingleOverlayStruct()` controlling a static overlay and for each `SingleOverlayStruct()` in a sample of an overlay timed metadata track with the content that is present in the sample or inherited from the sample entry of the overlay timed metadata track as specified in subclause 7.14.6.2:

— Controls with bit indices 0 to 3, inclusive, specify the rendering type of the overlay. In this version of this document, one and exactly one of `overlay_control_flag[i]` with `i` in the range of 0 to 3, inclusive, in each `SingleOverlayStruct()` shall be equal to 1.

— Controls with bit indices 4, 5, 6, and 14 specify the source of the overlay content. In this version of this document, one and exactly one of `overlay_control_flag[i]` with `i` equal to 4, 5, 6, or 14 in each `SingleOverlayStruct()` shall be equal to 1.

— At most one of `overlay_control_flag[8]` (overlay opacity) and `overlay_control_flag[13]` (overlay alpha composition) shall be equal to 1 in the same `SingleOverlayStruct()`.

7.14.3.2 Viewport-relative overlay

```
aligned(8) class ViewportRelativeOverlay() {
   unsigned int(16) rect_left_percent;
   unsigned int(16) rect_top_percent;
   unsigned int(16) rect_width_percent;
   unsigned int(16) rect_height_percent;
   unsigned int(5) media_alignment;
   unsigned int(1) relative_disparity_flag;
   bit(2) reserved = 0;
   if (relative_disparity_flag)
      signed int(16) disparity_in_percent;
   else
      signed int(16) disparity_in_pixels;
}
```

`rect_left_percent`, `rect_top_percent`, `rect_width_percent` and `rect_height_percent` specify the coordinates of the top-left corner and the width and height of the rectangular region, respectively, of the overlay to be rendered on the viewport in per cents relative to the width and height of the viewport. The values are indicated in units of 100% ÷ 2^{-16} in the range of 0 (indicating 0%), inclusive, up to but excluding 65536 (that indicates 100%).

NOTE The size and aspect ratio of the rectangular region on which the overlay is rendered changes according to the viewport resolution and aspect ratio. However, when the value of `media_alignment` is in the range 1 to 9, inclusive, the aspect ratio of the overlaid media is not intended to be changed.

`media_alignment` specifies the intended scaling of the overlay source depending on the dimensions of the specified rectangular region and the intended placement of the scaled overlay source relative to the specified rectangular region.

`media_alignment` equal to 0 specifies that an OMAF player should apply the "stretch to fill" mode, where the overlay source is horizontally and vertically scaled as needed to fill the specified rectangular region entirely. When `media_alignment` is equal to 0, the aspect ratio of the scaled overlay source may differ from the aspect ratio of the overlay source.

`media_alignment` values in the range of 1 to 9, inclusive, specify that an OMAF player should apply the "scale to fit" mode specified as follows: the overlay source is horizontally and vertically scaled to cover the specified rectangular region exactly along one dimension (i.e. either horizontally or vertically) and to cover or reside within the specified rectangular region along the other dimension while the aspect ratio of the overlay source is maintained. When the aspect ratio of the overlay source differs from the aspect ratio of the specified rectangular region, a part of the rectangular region is not filled.

`media_alignment` values in the range of 10 to 18, inclusive, specify that an OMAF player should apply the "crop to fill" mode specified as follows: the overlay source is horizontally and vertically scaled to cover the specified rectangular region exactly along one dimension (i.e. either horizontally or vertically) and to cover or exceed the specified rectangular region along the other dimension while the aspect ratio of the overlay source is maintained. An OMAF player should not display the parts of the scaled overlay source that exceed the specified rectangular region.

`media_alignment` values in the range of 19 to 31, inclusive, are reserved.

An OMAF player should place the scaled overlay source on the specified rectangular region as specified in Table 17.

Table 17 — Placement of the scaled overlay source on the specified rectangular region as a function of `media_alignment` values

Values	Description
1, 10	Horizontally and vertically centered
2, 11	Horizontally centered, vertically aligned at the top of the rectangular region
3, 12	Horizontally centered, vertically aligned at the bottom of the rectangular region
4, 13	Horizontally aligned with the left boundary of the rectangular region, vertically centered
5, 14	Horizontally aligned with the right boundary of the rectangular region, vertically centered
6, 15	Horizontally aligned with the left boundary of the rectangular region, vertically aligned at the top of the rectangular region
7, 16	Horizontally aligned with the right boundary of the rectangular region, vertically aligned at the top of the rectangular region
8, 17	Horizontally aligned with the left boundary of the rectangular region, vertically aligned at the bottom of the rectangular region
9, 18	Horizontally aligned with the right boundary of the rectangular region, vertically aligned at the bottom of the rectangular region

`relative_disparity_flag` indicates whether the disparity is provided as a percentage value of the width of the display window for one view (when the value is equal to 1) or as a number of pixels (when the value is equal to 0). This applies for the case when there is a monoscopic overlay.

`disparity_in_percent` indicates the disparity, in units of 2^{-16}, as a fraction of the width of the display window for one view. The value may be negative, in which case the displacement direction is reversed. This value is used to displace the region to the left on the left eye view and to the right on the right eye view. This applies for the case when there is a monoscopic overlay and stereoscopic background visual media.

`disparity_in_pixels` indicates the disparity in pixels. The value may be negative, in which case the displacement direction is reversed. This value is used to displace the region to the left on the left eye view and to the right on the right eye view. This applies for the case when there is a monoscopic overlay and stereoscopic background visual media.

7.14.3.3 Sphere-relative projected omnidirectional overlay

```
aligned(8) class SphereRelativeOmniOverlay() {
   unsigned int(1) region_indication_type;
   unsigned int(1) timeline_change_flag;
   bit(6) reserved = 0;
   if (region_indication_type == 0) {
      unsigned int(32) proj_picture_width;
      unsigned int(32) proj_picture_height;
      unsigned int(32) proj_reg_width;
      unsigned int(32) proj_reg_height;
      unsigned int(32) proj_reg_top;
      unsigned int(32) proj_reg_left;
   }
   else
      SphereRegionStruct(1, 1);
   unsigned int(16) region_depth_minus1;
}
```

SphereRelativeOmniOverlay shall not be present when the background visual media uses the 3D mesh.

region_indication_type equal to 0 specifies that the sphere region on which the overlay is rendered is indicated as a rectangle in a projected picture. region_indication_type equal to 1 specifies that the sphere region on which the overlay is rendered is indicated by the SphereRegionStruct() included in this syntax structure. region_indication_type shall be equal to 0 when the projection format of the overlay and the background visual media is identical.

When the SphereRegionStruct() is present (i.e. region_indication_type is equal to 1), the sphere region represented by SphereRegionStruct() shall be specified by two azimuth circles and two elevation circles and the associated projection format shall be ERP.

When the SphereRegionStruct() is present (i.e. region_indication_type is equal to 1), interpolate shall be equal to 0.

When region_indication_type is equal to 0, the syntax structure indicates that the projected pictures of overlays are packed region-wise and require unpacking prior to rendering, according to the region-wise packing process information as indicated. The semantics of the syntax elements are identical to those for the syntax elements of RegionWisePackingStruct( ) with the same name as specified in subclause 7.5.3.

The ProjectionFormatBox, StereoVideoBox, and RotationBox indirectly contained in the same sample entry that directly or indirectly contains the OverlayConfigBox containing this OverlayStruct() apply to the projected omnidirectional overlay(s) contained in the track. The ProjectionFormatProperty, FramePackingProperty, and RotationProperty associated with the image item that is also associated with the OverlayProperty containing this OverlayStruct() apply to the projected omnidirectional overlay(s).

NOTE In order to render a sphere-relative omnidirectional overlay without deformations, the azimuth and elevation ranges of the overlay source need to be equally large as the azimuth and elevation ranges of the region on which the overlay is rendered.

timeline_change_flag equal to 1 specifies that the overlay content playback shall pause if the overlay is not in the user's current viewport, and when the overlay is back in the user's viewport the overlay content playback shall resume with the global presentation timeline of the content. The content in the intermediate interval is skipped. timeline_change_flag shall be equal to 1, when the overlay source is specified by OverlaySourceRegion. timeline_change_flag equal to 0 specifies that the overlay content playback shall pause if the overlay is not in the user's current viewport, and when the overlay is back in the user's viewport the overlay content playback resumes from the paused sample. This prevents loss of any content due to the overlay being away from the user's current viewport.

region_depth_minus1 indicates the depth (z-value) of the region on which the overlay is to be rendered. The depth value is the norm of the normal vector of the overlay region. region_depth_minus1 + 1 specifies the depth value relative to a unit sphere in units of 2^{-16}. When region_depth_minus1 is equal to 65535 in SphereRelativeOmniOverlay, the overlay source and the background visual media shall be associated with the same projection format and the same yaw, pitch, and roll angles of the rotation to be applied to convert the local coordinate axes to the global coordinate axes.

7.14.3.4 Sphere-relative two-dimensional overlay

```
aligned(8) class SphereRelative2DOverlay() {
   SphereRegionStruct(1, 1);
   unsigned int(1) timeline_change_flag;
   bit(7) reserved = 0;
   signed int(32) overlay_rot_yaw;
   signed int(32) overlay_rot_pitch;
   signed int(32) overlay_rot_roll;
   unsigned int(16) region_depth_minus1;
}
```

The syntax elements of the SphereRegionStruct(1, 1) contained within the SphereRelative2DOverlay() syntax structure have the following semantics:

— centre_azimuth and centre_elevation specify the azimuth and elevation angles, respectively, of the centre of the overlay region on the unit sphere relative to the global coordinate axes.

— centre_tilt shall be equal to 0.

— azimuth_range and elevation_range specify the dimensions of the 2D plane on which the overlay is rendered. The sphere region represented by SphereRegionStruct() is specified by four great circles.

NOTE 1 Prior to rendering the 2D plane, it can be rotated as specified by overlay_rot_yaw, overlay_rot_pitch and overlay_rot_yaw and placed on a certain distance as specified by region_depth_minus1.

— interpolate shall be equal to 0.

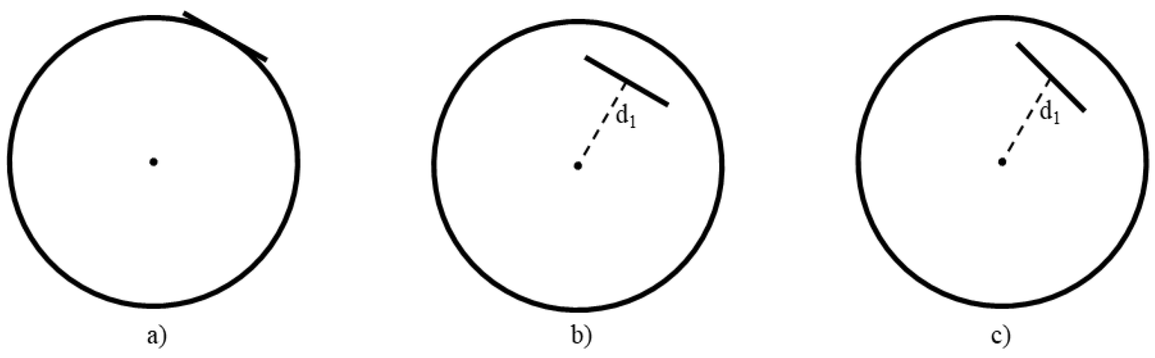
timeline_change_flag is specified identically to the syntax element with the same name in SphereRelativeOmniOverlay.

overlay_rot_yaw, overlay_rot_pitch, and overlay_rot_roll specify the rotation of the plane on which the overlay is rendered. The rotations are relative to the coordinate system as specified in subclause 5.1 in which the origin of the coordinate system is in the centre of the overlay region, the X axis is towards the origin of the global coordinate axes, the Y axis is towards the point on the plane that corresponds to cAzimuth1 in Figure 20, and the Z axis is towards the point on the plane that corresponds to cElevation2 in Figure 20. overlay_rot_yaw expresses a rotation around the Z axis, overlay_rot_pitch rotates around the Y axis, and overlay_rot_roll rotates around the X axis. Rotations are extrinsic, i.e. around X, Y, and Z fixed reference axes. The angles increase clockwise when looking from the origin towards the positive end of an axis. The rotations are applied starting from overlay_rot_yaw, followed by overlay_rot_pitch, and ending with overlay_rot_roll.

NOTE 2 The specification of the rotations with overlay_rot_yaw, overlay_rot_pitch, and overlay_rot_roll is aligned with the specifications of subclause 5.3.

region_depth_minus1 is specified identically to the syntax element with the same name in SphereRelativeOmniOverlay.

NOTE 3 Figure 25 provides an example how the placement and orientation of the 2D plane for the sphere-relative 2D overlay is determined based on the syntax elements of SphereRelative2DOverlay.



Key

a) 2D plane based on the indicated sphere region

b) moving the 2D plane to an indicated depth d$_1$

c) rotating the 2D plane as indicated in `SphereRelative2DOverlay`

Figure 25 — Example of placement and orientation of sphere-relative 2D overlay, illustrated with an equator-level cross section of the unit sphere

7.14.3.5 3D mesh overlay

```
aligned(8) class MeshOverlay() {
   unsigned int(1) timeline_change_flag;
   unsigned int(1) eye_present_flag;
   bit(6) reserved = 0;
   unsigned int(8) num_regions;
   for (i = 0; i < num_regions; i++) {
      if (eye_present_flag) {
         unsigned int(1) eye[i];
         bit(7) reserved = 0;
      }
      unsigned int(16) mesh_element_id[i];
      RectRegionStruct();
   }
}
```

`timeline_change_flag` is specified identically to the syntax element with the same name in `SphereRelativeOmniOverlay`.

The semantics of the other syntax elements are identical to those for the syntax elements with the same name in `TileMeshGroupEntry` specified in subclause 7.16.2.

7.14.3.6 Source region for the overlay

```
aligned(8) class OverlaySourceRegion() {
   unsigned int(16) packed_picture_width;
   unsigned int(16) packed_picture_height;
   unsigned int(16) packed_reg_width;
   unsigned int(16) packed_reg_height;
   unsigned int(16) packed_reg_top;
   unsigned int(16) packed_reg_left;
   unsigned int(3)  transform_type;
   bit(5) reserved = 0;
}
```

The syntax structure indicates the source region of the overlay within the decoded picture.

If the track or image item containing the overlay is a projected omnidirectional video track or image item, the semantics of the syntax elements are identical to those for the syntax elements of `RegionWisePackingStruct()` with the same name as specified in subclause 7.5.3. Otherwise, the semantics of the syntax elements are specified as follows:

`packed_picture_width` and `packed_picture_height` specify the width and height of the decoded picture, respectively, in luma samples.

`packed_reg_width`, `packed_reg_height`, `packed_reg_top`, and `packed_reg_left` specify the width, height, top offset, left offset of the overlay source, respectively, in luma samples within the decoded picture. When the overlay is included in the same track or image item with background visual media, the overlay source region shall not be overlapped with any packed regions of the background visual media.

`transform_type` specifies the rotation and mirroring that are applied to the overlay source to remap it for displaying. The following values are specified:

0: no transform

1: mirroring horizontally

2: rotation by 180 degrees (counter-clockwise)

3: rotation by 180 degrees (counter-clockwise) before mirroring horizontally

4: rotation by 90 degrees (counter-clockwise) before mirroring horizontally

5: rotation by 90 degrees (counter-clockwise)

6: rotation by 270 degrees (counter-clockwise) before mirroring horizontally

7: rotation by 270 degrees (counter-clockwise)

NOTE 1 When the track or image item containing the overlay is not a projected omnidirectional video track or image item, the OMAF player can treat each decoded picture as a texture atlas including sub-images indicated by the overlay source region information. The OMAF player does not need to perform an unpacking process, similar to what is specified for `RegionWisePackingStruct()`, for processing of such a texture atlas.

NOTE 2 When the track or image item containing the overlay is a projected omnidirectional video track or image item that also contains the background visual media, the overlay and the background visual media have the same projection format and the same the yaw, pitch, and roll angles of the rotation to be applied to convert the local coordinate axes to the global coordinate axes.

7.14.3.7 Overlay layering order

```
aligned(8) class OverlayLayeringOrder() {
   int(16) layering_order;
}
```

`layering_order` indicates the layering order among the overlays that are relative to the viewport, and separately among each set of overlays that have the same depth. Viewport-relative overlays are overlaid on top of the viewport in descending order of `layering_order`, i.e. an overlay with a smaller `layering_order` value is in front of an overlay with a greater `layering_order` value. For sphere-relative overlays, the smaller the value of `layering_order`, the closer the overlay is from the viewer. When not present, `layering_order` is inferred to be equal to 0.

NOTE It is unspecified how to render two overlays that either have the same depth or are both viewport-relative overlays, have the same `layering_order` value, and are overlapping.

7.14.3.8 Overlay opacity

The overlay opacity affects the rendering identically to the source over composition mode, specified in subclause 7.14.3.13, that is applied to an alpha plane in which all pixels have a constant value.

```
aligned(8) class OverlayOpacity() {
   unsigned int(8) opacity;
}
```

opacity is an integer value that specifies the opacity that is to be applied for the overlay. Value 0 is fully transparent and value 100 is fully opaque. Values greater than 100 are reserved.

7.14.3.9 Controls for user interaction

```
aligned(8) class OverlayInteraction() {
   unsigned int(1) change_position_flag;
   unsigned int(1) change_depth_flag;
   unsigned int(1) switch_on_off_flag;
   unsigned int(1) change_opacity_flag;
   unsigned int(1) resize_flag;
   unsigned int(1) rotation_flag;
   unsigned int(1) source_switching_flag;
   unsigned int(1) crop_flag;
}
```

change_position_flag, when set to 1, specifies that users are allowed to move the overlay window to any location on the viewing sphere or the viewport.

change_depth_flag, when set to 1, specifies that the depth of overlay can be chosen by user interaction. When both change_position_flag and change_depth_flag are set to 1 then the X,Y,Z position of the overlay can be freely choosen by user interaction.

switch_on_off_flag, when set to 1, specifies that the user is allowed to switch ON/OFF the overlay. When the AssociatedSphereRegion structure is present for the overlay, the value of the switch_on_off_flag shall be equal to 1. All active overlays are considered to be ON by default. When an overlay is switched ON, the playback position for the overlay is the current time on the global presentation timeline.

change_opacity_flag, when set to 1, specifies that the user is allowed to change the opacity of the overlay.

resize_flag, when set to 1, specifies that the user is allowed to resize the overlay window. The field-of-view of the resized overlay window shall be same as that of original overlay window.

rotation_flag, when set to 1, specifies that the user is allowed to rotate the overlay window to different directions. The field-of-view of the rotated overlay window shall be same as that of original overlay window.

source_switching_flag,when set to 1, specifies that the user is allowed to switch to a new overlay source from the current overlay source. The new overlay source that the user switches to shall be part of the same entity group as the current overlay source with the grouping_type equal to 'oval', as specified in subclause 7.14.7.1.

crop_flag equal to 1 specifies that user is allowed to crop the overlay media window.

change_position_flag, change_depth_flag, switch_on_off_flag, change_opacity_flag, resize_flag, rotation_flag, source_switching_flag, or crop_flag when set to 0, specifies that the user is disallowed to perform the respective operation on the overlay.

change_position_flag, change_depth_flag, change_opacity_flag, resize_flag, rotation_flag, and crop_flag shall be equal to 0 for all the overlays in the same 'oval' entity group.

7.14.3.10 Overlay label

```
aligned(8) class OverlayLabel() {
   string overlay_label;
}
```

`overlay_label` provides a null-terminated UTF-8 label for the overlay.

7.14.3.11 Overlay priority

```
aligned(8) class OverlayPriority() {
   unsigned int(8) overlay_priority;
}
```

`overlay_priority` indicates which overlay should be prioritized in the case that an OMAF player does not have enough decoding capacity to decode all overlays. A lower `overlay_priority` indicates higher priority. A value of `overlay_priority` equal to 255 indicates unspecified priority.

The value of `overlay_priority`, when present, shall be equal to 0 for overlays that are essential for displaying.

7.14.3.12 Associated sphere region

```
aligned(8) class AssociatedSphereRegion() {
   unsigned int(8) shape_type;
   SphereRegionStruct(1, 1);
}
```

`shape_type` is used as the shape type value when applying subclause 7.5.6 to the semantics of the `SphereRegionStruct()` specifying the associated sphere region.

`SphereRegionStruct(1, 1)` specifies the sphere region associated with the overlay. The sphere region associated with an overlay is intended to be used to turn on or turn off the overlay.

When more than one overlay is associated with the same sphere region, the overlays should all be displayed when the user selects the sphere region.

NOTE When multiple associated sphere regions fully or partially overlap and the user selects the overlapping area of the associated sphere regions, the OMAF player is expected to turn on/off all the overlays controlled by all the overlapping associated sphere regions.

For the `SphereRegionStruct(1, 1)` included in `AssociatedSphereRegion`, `interpolate` shall be equal to 0.

7.14.3.13 Overlay alpha composition

```
aligned(8) class OverlayAlphaCompositing() {
   unsigned int(8) alpha_blending_mode;
}
```

`alpha_blending_mode` specifies the alpha blending mode as listed in Table 18. When an overlay is associated with an alpha plane as specified in subclause 7.14.8, the `alpha_blending_mode` indicates how to compose the overlay.

Table 18 — Alpha blending modes

Value	Composition mode	Description	Formula
0	Source over	Source is placed over the destination.	$v_u = m \times \alpha + v_i \times (1 - \alpha)$
1..255	Reserved	For new modes or non-separable blending modes	

NOTE The formula in Table 18 is the same as in ISO/IEC 23000-22 for overlay images that are not pre-multiplied.

In the formula in Table 18, v_u is a pixel value in the updated visual context, m is a pixel value in the associated overlay, α is an alpha plane value, scaled into the range of 0 (fully transparent) to 1 (fully opaque), inclusive, and v_i is a pixel value in the background visual context given as input to the process. The visual context is defined as visual rendering surface or mesh, such as a viewport, which initially contains the background visual media and has been updated by all overlays that are further away from the viewer in distance or layering order than the associated overlay.

7.14.3.14 Externally specified overlay information

```
aligned(8) class ExternallySpecifiedOverlayInfo() {
   utf8string externally_specified_overlay_scheme_URI;
   unsigned int(8) num_scheme_bytes;
   for (i = 0; i < num_scheme_bytes; i++)
      unsigned int(8) scheme_byte_val[i];
}
```

externally_specified_overlay_scheme_URI is a null terminated UTF-8 string that indicates the URI specifying a scheme to interpret the value signalled in scheme_byte_val[i] syntax elements. externally_specified_overlay_scheme_URI shall be a valid URI as specified in IETF Internet Standard 66. When an ExternallySpecifiedOverlayInfo structure is present, externally_specified_overlay_scheme_URI shall not be null.

NOTE Examples of how externally specified overlay information can be used include the following:

— externally_specified_overlay_scheme_URI can indicate that a webcam data source to be used as an overlay source.

— externally_specified_overlay_scheme_URI can be a URL to an image used as an overlay source.

— externally_specified_overlay_scheme_URI and the array of scheme_byte_val[i] can together indicate both an externally specified overlay source that is a logo image or video and a 3D spinning effect applied to the logo.

num_scheme_bytes specifies the number of bytes (scheme_byte_val[i] bytes) that follow in this overlay control structure. num_scheme_bytes may be equal to 0.

scheme_byte_val[i] is i-th byte of data which is interpreted based on the scheme signalled in externally_specified_overlay_scheme_URI. scheme_byte_val[i] is absent when num_scheme_bytes is equal to 0.

7.14.4 Overlay configuration box

Box type:	'ovly'
Container:	ProjectedOmniVideoBox or VisualSampleEntry or MetaDataSampleEntry
Mandatory:	Yes, in a media track that contains overlays and in an overlay timed metadata track and in a recommended viewport timed metadata track specifying an overlay source
Quantity:	Zero or one

OverlayConfigBox is defined to store the static metadata of the overlays contained in this track.

When a track has a sample entry type equal to 'resv' and scheme_type equal to 'podv' in the SchemeTypeBox included in the sample entry, the track contains background visual media, and the track contains one or more overlays with overlay_priority equal to 0, OverlayConfigBox shall be contained directly within the ProjectedOmniVideoBox.

When a recommended viewport track is used to specify an overlay source, OverlayConfigBox shall be contained within the MetaDataSampleEntry of the recommended viewport timed metadata track.

When an OverlayConfigBox is present in ProjectedOmniVideoBox or VisualSampleEntry of a media track or in MetaDataSampleEntry of the recommended viewport timed metadata track, it shall contain all the overlay_id values of the overlays carried at any time in the media track or the recommended viewport timed metadata track, respectively.

When an overlay is controlled through an overlay timed metadata track, the OverlayConfigBox that is present in ProjectedOmniVideoBox or VisualSampleEntry of a media track or in MetaDataSampleEntry of the recommended viewport timed metadata track, if any, should either contain no overlay_control_flag[i] syntax elements for that overlay or have overlay_control_flag[i] equal to 0 for all values of i for that overlay.

NOTE When an overlay timed metadata track is in use, all the overlay controls are provided through the sample entry or samples of the overlay timed metadata track.

```
aligned(8) class OverlayConfigBox extends FullBox('ovly', 0, 0) {
   OverlayStruct();
}
```

7.14.5 Overlay item property

Box type:	'ovly'
Property type:	Descriptive item property
Container:	ItemPropertyContainerBox
Mandatory:	No
Quantity:	Zero or one

OverlayConfigProperty is defined to store the static metadata of the overlays contained in the associated image item.

```
aligned(8) class OverlayConfigProperty extends ItemFullProperty ('ovly', 0, 0) {
   OverlayStruct();
}
```

7.14.6 Overlay timed metadata track

7.14.6.1 Definition

The dynamic overlay timed metadata track indicates:

— Which overlays are active at particular times. Depending upon the application the active overlay(s) may change over time.

— Overlay parameters that may be dynamically changing over time.

The overlay timed metadata track is linked to the respective overlay visual media tracks by utilizing the 'cdsc' track reference.

7.14.6.2 Sample entry

```
aligned(8) class OverlaySampleEntry extends MetadataSampleEntry('dyol') {
   OverlayConfigBox();
}
```

The sample entry of an overlay timed metadata track contains an `OverlayConfigBox` that includes the default syntax element values of `OverlayStruct()` that apply selectively when both of the following conditions are true:

— The same `overlay_id` is present in a sample.

— When `byte_count[i]` is present and equal to 0 for a particular `overlay_id` in the `OverlayStruct()` of an overlay timed metadata sample and `byte_count[i]` is present and greater than 0 for the same `overlay_id` in the `OverlayStruct()` of the sample entry, `overlay_control_struct[i][byte_count[i]]` of the sample entry and for the same `overlay_id` value applies.

7.14.6.3 Sample

```
aligned(8) class OverlaySample {
   unsigned int(15) num_active_overlays_by_id;
   unsigned int(1)  addl_active_overlays_flag;
   for (i = 0; i < num_active_overlays_by_id; i++)
      unsigned int(16) active_overlay_id;
   if(addl_active_overlays_flag)
      OverlayStruct();
}
```

`num_active_overlays_by_id` specifies the number of overlays from the `OverlayStruct()` structure signalled in the sample entry `OverlaySampleEntry` that are active. A value of 0 indicates that no overlays from the sample entry are active.

`addl_active_overlays_flag` equal to 1 specifies that additional active overlays are signalled in the sample directly in the overlay structure (`OverlayStruct()`). `addl_active_overlays_flag` equal to 0 specifies that no additional active overlays are signalled in the sample directly in the overlay structure (`OverlayStruct()`). Each `overlay_id` value contained in the `OverlayStruct()` of an `OverlaySample()` shall be equal to an `overlay_id` value contained in the `OverlaySampleEntry` of the same overlay timed metadata track and to an `overlay_id` value in the `OverlayConfigBox` contained in one of the media tracks referenced by this overlay timed metadata track.

`active_overlay_id` provides overlay identifier for the overlay signalled from the sample entry, which is currently active. For each `active_overlay_id`, the `OverlayStruct()` structure in the sample entry `OverlaySampleEntry` shall include an overlay with a matching `overlay_id` value. `active_overlay_id` shall be equal to an `overlay_id` value in the `OverlayConfigBox` contained in one of the media tracks referenced by this overlay timed metadata track.

`num_overlays` of a sample is not required to be equal to `num_overlays` in the sample entry, and the set of `overlay_id` values of a sample is not required to be the same as the set of `overlay_id` values in the sample entry.

Activation of particular overlays by a sample results in deactivation of any previously signalled overlays from previous sample(s).

7.14.7 Entity groups

7.14.7.1 Grouping of overlays that are alternatives for switching

7.14.7.1.1 Definition

`EntityToGroupBox` with `grouping_type` equal to `'oval'` specifies tracks and image items containing overlays intended to be presented as a user-switchable alternative for another overlay in the same entity group.

7.14.7.1.2 Syntax

```
aligned(8) class OverlaySwitchAlternativesBox(version, flags)
extends EntityToGroupBox('oval', version, flags) {
    // conditionally mandatory
    for(i=0; i<num_entities_in_group; i++)
        unsigned int(16) ref_overlay_id[i];
}
```

7.14.7.1.3 Semantics

`ref_overlay_id[i]` specifies the `overlay_id` from the track or image item identified by i-th `entity_id` that is a switchable overlay in this group. The overlay identified by `ref_overlay_id[0]` is the initial overlay source, before any user interaction. The i-th referenced track or image item shall have `overlay_id` equal to `ref_overlay_id[i]` present. If each of the tracks and image items identified by the `entity_id` values of this entity group contains exactly one overlay, `ref_overlay_id[i]` syntax elements may or may not be present. Otherwise, `ref_overlay_id[i]` syntax elements shall be present.

7.14.7.2 Grouping of overlays and background visual media that are intended to be presented together

7.14.7.2.1 Definition

`EntityToGroupBox` with `grouping_type` equal to `'ovbg'` specifies

— tracks and image items containing overlays and background visual media that are intended to be presented together,

— recommended viewport tracks that serve as overlay sources for the background visual media specified in this entity group,

— at most one overlay timed metadata track that applies when playing the overlays and the background visual media included in this entity group, and

— at most one viewing space (`'vrsp'`) timed metadata track that applies when playing the overlays and the background visual media included in this entity group.

If the i-th entity in the `'ovbg'` entity group includes overlays, `overlay_flag[i]` shall be equal to 1. Otherwise, `overlay_flag[i]` shall be equal to 0.

NOTE 1 The presence of overlays can also be determined as follows: When a track in an `'ovbg'` entity group contains an `OverlayConfigBox` in its sample entry, it includes overlays. When an image item in an `'ovbg'` entity group is associated with an overlay item property (i.e. `OverlayConfigProperty`), it includes overlays.

The pair of `overlay_flag[i]` equal to 1 and `overlay_subset_flag[i]` equal to 0 specifies that all the overlays specified in the `i`-th entity are associated with the background visual media specified in this entity group. When `overlay_subset_flag[i]` is equal to 1, the `'ovbg'` entity group contains a list of overlay ID values of the overlays in the `i`-th entity that are associated with the background visual media specified in this entity group.

If the i-th entity in the 'ovbg' entity group includes background visual media, backgound_flag[i] shall be equal to 1. Otherwise, background_flag[i] shall be equal to 0.

NOTE 2 The presence of background visual media without overlays can also be determined as follows. When a track in an 'ovbg' entity group does not contain an OverlayConfigBox in its sample entry, it is a background visual media track without overlays. When an image item in an 'ovbg' entity group is not associated with an overlay item property (i.e. OverlayConfigProperty), it is a background visual image item without overlays.

NOTE 3 When both overlay_flag[i] and background_flag[i] are equal to 1 for the same value of i, both background visual media and overlays are present in the i-th entity.

The background visual media tracks and the background image items in the same 'ovbg' entity group shall be constrained in either of the following ways:

— All the background visual media tracks and background image items are alternatives to each other, indicated by their presence in the same 'altr' entity group.

— There are no background image items in the 'ovbg' entity group and the background visual media tracks in the 'ovbg' entity group belong to the same 2D spatial relationship track group.

When both one or more overlays and background visual media are region-wise packed into the same video track or image item included in an 'ovbg' entity group, the same 'ovbg' entity group shall contain no other track or image item containing background visual media.

When a user-switchable overlay included in an 'oval' entity group is associated with background visual media in an 'ovbg' entity group, all the overlays of the 'oval' entity group shall be associated with the background visual media in the 'ovbg' entity group.

7.14.7.2.2 Syntax

```
aligned(8) class OverlayAndBackgroundGroupingBox(version, flags)
extends EntityToGroupBox('ovbg', version, flags) {
   unsigned int(32) unit_sphere_distance_in_mm;
   for (i=0; i<num_entities_in_group; i++) {
      bit(5) reserved = 0;
      unsigned int(1) overlay_flag[i];
      unsigned int(1) overlay_subset_flag[i];
      unsigned int(1) background_flag[i];
      if (overlay_subset_flag[i]) {
         unsigned int(16) ovbg_num_overlays;
         for (j=0; j<ovbg_num_overlays; j++)
      unsigned int(16) ovbg_overlay_id[i][j];
      }
   }
}
```

7.14.7.2.3 Semantics

unit_sphere_distance_in_mm, when greater than 0, specifies a distance, in millimetres, corresponding to the radius of the unit sphere. The value should be used in the following cases:

— For stereoscopic rendering of the content on the unit sphere together with overlaying content and for deriving suitable binocular disparity for overlaying visual tracks or image items for which the depth is indicated relative to the unit sphere.

— In a rendering procedure that takes the viewing position into account, for handling the changes of the viewing position relative to the overlays and the background visual media in the same real-world scale.

unit_sphere_distance_in_mm equal to 0 specifies that the unit sphere distance is unknown.

NOTE unit_sphere_distance_in_mm is applicable for stereoscopic rendering when the background visual media or the overlay is monoscopic or when both the background visual media and the overlay are monoscopic.

overlay_flag[i] equal to 0 specifies that the entity does not contain any overlays. overlay_flag[i] equal to 1 specifies that the entity contains one or more overlays.

background_flag[i] equal to 0 specifies that the entity does not contain background visual media. background_flag[i] equal to 1 specifies that the entity contains background visual media.

One or both of overlay_flag[i] and background_flag[i] shall be equal to 1 for each value of i in the range of 0 to num_entities_in_group − 1, inclusive, such that the i-th entity_id value is a track_ID value of a media track or an item_ID value of an image item.

The pair of overlay_flag[i] equal to 1 and overlay_subset_flag[i] equal to 0 specifies that all the overlays specified in the i-th entity are associated with the background visual media specified in this entity group. overlay_subset_flag[i] equal to 1 specifies that some but not all of the overlays specified in the i-th entity are associated with the background visual media specified in this entity group. When overlay_flag[i] is equal to 0, overlay_subset_flag[i] shall be equal to 0. The semantics of the pair of overlay_flag[i] equal to 0 and overlay_subset_flag[i] equal to 0 are reserved.

ovbg_num_overlays specifies the number of overlays in the i-th entity that are associated with the background visual media specified in this entity group. ovbg_num_overlays shall not be equal to 0.

ovbg_overlay_id[i][j] specifies the overlay ID of the j-th overlay in the i-th entity that is associated with the background visual media specified in this entity group. When overlay_subset_flag[i] is equal to 1, all overlays in the i-th entity with an overlay ID not among any ovbg_overlay_id[i][j] are consistently inactive with the background visual media specified in this entity group regardless of any other overlay controls.

7.14.8 Overlay alpha auxiliary image

7.14.8.1 General

An alpha plane is an image of the same width and height as the master image, specifying the transparency information of the master image as defined in ISO/IEC 23008-12. An alpha plane is an auxiliary image stored as an image item or a sample of a video track.

7.14.8.2 Alpha auxiliary image constraints

The constraints provided in this subclause are applicable to auxiliary images stored as image items, as defined in ISO/IEC 23000-22:2019, subclause 7.3.5.2.

The following constraints apply to alpha planes:

— The width and height of alpha plane shall be the same as the width and height of the master overlay image.

— The minimum sample value 0 means that the co-located pixel in the master overlay image is transparent (i.e. won't be displayed).

— The maximum sample value (e.g. 255 for 8-bit sample values) means that the co-located pixel in the master overlay image is opaque.

— Each sample value of the alpha plane shall be in the range of 0 to the maximum value (e.g. 255 for 8-bit sample values), inclusive.

— The sample values of the alpha mask divided by the maximum value (e.g. by 255 in 8-bit) provides the multiplier to be used to obtain the intensity for the associated overlay image.

The master image(s) is not pre-multiplied by the sample values of the alpha plane.

As defined in ISO/IEC 23000-22, the value of `aux_type` for images of the `AuxiliaryTypeProperty` is as `"urn:mpeg:mpegB:cicp:systems:auxiliary:alpha"`.

An alpha auxiliary image and a master overlay image are linked using an item reference of `'auxl'` from the auxiliary image to the master overlay image as defined in ISO/IEC 23008-12.

7.14.8.3 Alpha auxiliary video track constraints

The constraints provided in this subclause are applicable to auxiliary images stored as samples of a video track, as defined in ISO/IEC 23000-22:2019, subclause 7.4.6. The constraints specified in ISO/IEC 23000-22:2019, subclause 7.4.6 apply.

As defined in ISO/IEC 23000-22, the value of `aux_track_type` for tracks of the `AuxiliaryTypeBox` is `"urn:mpeg:mpegB:cicp:systems:auxiliary:alpha"` for alpha planes.

The auxiliary and the master overlay video tracks are linked using the track reference of `'auxl'` from the auxiliary video track to the master video track as documented in ISO/IEC 23000-22.

7.14.8.4 Alpha auxiliary image/image sequence encoding

Alpha planes may be encoded in monochrome or colour. When an alpha plane is encoded in a colour format with a luma plane and chroma planes, e.g. as 4:2:0 YCbCr, only the luma plane is relevant, and the chroma planes shall be ignored by the OMAF player.

7.15 Signalling of viewing space information

7.15.1 General

The signalling of viewing spaces specified in subclause 7.15 only applies for overlays.

Viewing space is 3D space of viewing positions within which rendering of image and video is enabled and VR experience is valid.

A guard range indicates the 3D space within the viewing space and adjacent to the boundaries of the viewing space where the viewing experience will not be optimal when the viewing position is taken into account in rendering. An OMAF player may use visual cues to indicate when a viewing position is within the guard range. For example, an OMAF player can fade the viewport towards black within the guard range.

When viewpoint position information is present, the centre position of the viewing space is equal to the position of the viewpoint. When viewpoint position information is not present, the centre position of the viewing space is equal to (0,0,0) in the common reference coordinate system. In both cases, the X, Y, and Z coordinate axes are aligned with the reference global coordinate axes.

7.15.2 Viewing space structure

7.15.2.1 Definition

`ViewingSpaceStruct()` specifies the viewing space metadata information.

7.15.2.2 Syntax

```
aligned(8) class ViewingSpaceStruct() {
   unsigned int(8) viewing_space_shape_type;
   unsigned int(16) distance_scale;
   bit(1) guard_range_flag;
   bit(7) reserved;
   if(viewing_space_shape_type==0)
      CuboidStruct(guard_range_flag);
   else if(viewing_space_shape_type==1)
      SphereStruct(guard_range_flag);
   else if(viewing_space_shape_type==2)
      CylinderStruct(guard_range_flag);
   else if(viewing_space_shape_type==3)
      EllipsoidStruct(guard_range_flag);
}

aligned(8) class CuboidStruct(guard_range_flag) {
   signed int(32) x_Min;
   signed int(32) x_Max;
   signed int(32) y_Min;
   signed int(32) y_Max;
   signed int(32) z_Min;
   signed int(32) z_Max;
   if (guard_range_flag) {
      unsigned int(8) guard_range_X;
      unsigned int(8) guard_range_Y;
      unsigned int(8) guard_range_Z;
   }
}

aligned(8) class SphereStruct(guard_range_flag) {
   unsigned int (32) sphere_radius;
   if (guard_range_flag){
      bit(1) reserved;
      unsigned int(7) guard_radius_diff;
   }
}

aligned(8) class CylinderStruct(guard_range_flag) {
   unsigned int (32) cylinder_radius;
   Point(0);
   Point(1);
   if (guard_range_flag)
      unsigned int(8) cylinder_guard_radius_diff;
}

aligned(8) class Point(i) {
   signed int(32) x_pt[i];
   signed int(32) y_pt[i];
   signed int(32) z_pt[i];
}
```

```
aligned(8) class EllipsoidStruct(guard_range_flag) {
   unsigned int (32) length_X;
   unsigned int (32) length_Y;
   unsigned int (32) length_Z;
   if (guard_range_flag) {
      unsigned int(8) guard_lenghthX_diff;
      unsigned int(8) guard_lenghthY_diff;
      unsigned int(8) guard_lenghthZ_diff;
   }
}
```

7.15.2.3 Semantics

viewing_space_shape_type specifies the shape of the viewing space. viewing_space_shape_type equal to 0 specifies that the viewing space is specified as a cuboid. viewing_space_shape_type equal to 1 specifies that the viewing space is specified as a sphere. viewing_space_shape_type equal to 2 specifies that the viewing space is specified as a cylinder. viewing_space_shape_type equal to 3 specifies that the viewing space is specified as an ellipsoid. The value of viewing_space_shape_type shall be in the range of 0 to 3, inclusive.

distance_scale is a positive integer value that indicates the units corresponding to 1 cm.

guard_range_flag equal to 0 specifies that guard range information is not present in the ViewingSpaceStruct(). guard_range_flag equal to 1 specifies that guard range information is present in the ViewingSpaceStruct().

guard_range_X, guard_range_Y and guard_range_Z specify the guard ranges per axis as percentage of the maximum viewing space ranges per axis, i.e. Abs(xMax − xMin), Abs(yMax − yMin) and Abs(zMax − zMin) respectively. Guard ranges apply symmetrically to both minimum and maximum values per axis as indicated by CuboidStruct(). A value of 0 specifies that there is no guard range present for a particular axis. Values of guard_range_X, guard_range_Y and guard_range_Z shall be less than 50.

x_Min, y_Min, and z_Min are 16.16 fixed-point values in distance scale that specify the minimum value of X, Y, and Z coordinates, respectively, relative to the centre point of the viewing space.

x_Max, y_Max, and z_Max are 16.16 fixed-point values in distance scale that specify the maximum value of X, Y, and Z coordinates, respectively, relative to the centre point of the viewing space. x_Max shall be greater than x_Min. y_Max shall be greater than y_Min. z_Max shall be greater than z_Min.

sphere_radius specifies the radius of a sphere as a 16.16 fixed-point value in 3D space in distance scale. Value 0 is reserved.

guard_radius_diff specifies the thickness of the guard range spherical shell inside the sphere of radius equal to radius as a percentage of the radius. guard_radius_diff equal to 0 indicates that the guard range is not present. guard_radius_diff shall be in the range of 0 to 99, inclusive. Values in the range of 100 to 127, inclusive, are reserved.

cylinder_radius specifies the radius of a cylinder in 3D space in suitable units with the cylinder formed around the line from Point(0) to Point(1). Value 0 is reserved.

cylinder_guard_radius_diff specifies the thickness of the guard range cylindrical shell inside the cylinder of radius equal to cylinder_radius as a percentage of the cylinder_radius. cylinder_guard_radius_diff equal to 0 indicates that the guard range is not present. cylinder_guard_radius_diff shall be in the range of 0 to 99, inclusive. Values in the range of 100 to 255, inclusive, are reserved.

x_pt[i], y_pt[i] and z_pt[i] are values in units of distance scale that specifies the X, Y, Z coordinates, respectively, of a point in 3D space with respect to the centre point of the viewing space.

length_X, length_Y, and length_Z specify respectively the semi-axes lengths of X, Y and Z axis of an ellipsoid that has the same centre as the viewing space, in units of 2^{-16} millimetres. length_X, length_Y, and length_Z shall be in the range of 1 to 65 536 * 2^{16} – 1 (i.e. 4 294 967 295), inclusive.

guard_lengthX_diff, guard_lengthY_diff and guard_lengthZ_diff specify the thickness of the guard range space between the outer ellipsoid with axis lengths in X, Y and Z equal to length_X, length_Y and length_Z, respectively, and the inner ellipsoid with axis lengths in X, Y and Z equal to 100 – guard_lengthX_diff, 100 – guard_lengthY_diff and 100 – guard_lengthZ_diff percentage of the specified length_X, length_Y and length_Z values, respectively. The values shall be in the range of 0 to 99, inclusive. Values in the range of 100 to 255, inclusive, are reserved.

7.15.3 Viewing space box

7.15.3.1 Definition

Box Type:	'vssn'
Container:	VisualSampleEntry (of the background visual media)
Mandatory:	No
Quantity:	Zero or one

The fields in this box specify 3D viewing space within which an immersive VR experience is provided.

7.15.3.2 Syntax

```
aligned(8) class ViewingSpaceBox extends FullBox('vssn', 0, flags) {
   ViewingSpaceStruct();
}
```

7.15.4 Viewing space item property

7.15.4.1 Definition

Box Type:	'vssn'
Property Type:	Descriptive item property
Container:	ItemPropertyContainerBox
Mandatory:	No
Quantity:	Zero or one

The fields in this item property specify 3D viewing space within which an immersive VR experience is provided. The item property shall be associated with an image item that contains the background visual media.

7.15.4.2 Syntax

```
aligned(8) class ViewingSpaceProperty extends ItemFullProperty ('vssn', 0, 0) {
   ViewingSpaceStruct();
}
```

7.15.5 Time varying immersive viewing space signalling

7.15.5.1 Definition

The immersive viewing space timed metadata track indicates the viewing space boundaries where immersive experience is supported. The space can be static or can change dynamically on sample basis.

A viewing space timed metadata track is associated with the background visual media and overlays specified by an 'ovbg' entity group by including it in the same 'ovbg' entity group.

7.15.5.2 Syntax and Semantics

The track sample entry type `'vrsp'` shall be used. The sample entry is specified as follows:

```
aligned(8) class VRSpaceSampleEntry(type) extends MetadataSampleEntry('vrsp') {
   unsigned int(1) static_vr_space_flag;
   if (static_vr_space_flag == 1)
      ViewingSpaceStruct();
}
```

`static_vr_space_flag` equal to 1 specifies that the immersive VR experience viewing space does not change for each sample referring to this sample entry. `static_vr_space_flag` equal to 0 specifies that the immersive 3D VR space may change for samples referring to this sample entry.

`ViewingSpaceStruct()` in the sample entry specifies the static viewing space.

The sample syntax is specified as follows:

```
aligned(8) class VRSpaceSample() {
   if(static_vr_space_flag == 0)
      ViewingSpaceStruct();
}
```

`ViewingSpaceStruct()` in the sample specifies the viewing space that is valid for the duration of the sample.

7.16 Mapping of rectangular regions to the 3D mesh

7.16.1 General

Subclause 7.16 specifies a sample group that maps rectangular regions of decoded pictures to mesh elements of a 3D mesh.

7.16.2 Tile mesh sample grouping

7.16.2.1 Definition

Group Type:	`'tmsh'`
Container:	`SampleGroupDescriptionBox ('sgpd')`
Mandatory:	No
Quantity:	Zero or one

The `TileMeshGroupEntry` describes one or more rectangular regions within the track, and describes how they map to one or more mesh elements described by the `MeshBox` associated with this track. The rectangular regions are defined relative to the decoded pictures of the current track.

7.16.2.2 Syntax

```
aligned(8) class TileMeshGroupEntry() extends VisualSampleGroupEntry ('tmsh') {
   unsigned int(8) num_regions;
   unsigned int(1) guard_band_flag;
   unsigned int(1) eye_present_flag;
   if (eye_present_flag == 0)
      unsigned int(1) common_eye;
   else
      bit(1) reserved = 0;
   bit(5) reserved = 0;
   for (i = 0; i < num_regions; i++) {
      if(eye_present_flag) {
         unsigned int(1) eye[i];
         bit(7) reserved = 0;
      }
      unsigned int(16) mesh_element_id[i];
      RectRegionStruct();
      if (guard_band_flag)
         GuardBand(i);
   }
}
```

7.16.2.3 Semantics

num_regions specifies the number of regions.

guard_band_flag specifies the existence of the GuardBand structure for all regions.

eye_present_flag equal to 1 specifies that syntax element eye[i] is present. eye_present_flag equal to 0 specifies that syntax element eye[i] is not present and is inferred. When the StereoVideoBox is not present in the SchemeInformationBox that contains the MeshBox associated with this sample group, eye_present_flag shall be equal to 0.

common_eye equal to 0, when present, specifies that all regions of this TileMeshGroupEntry correspond to the left eye. common_eye equal to 1, when present, specifies that all regions of this TileMeshGroupEntry correspond to the right eye.

eye[i] specifies the eye associated with the i-th rectangular region. When set to 0, the region corresponds to the left eye; when set to 1, the region corresponds to the right eye. When not present, eye[i] is inferred to be equal to common_eye.

mesh_element_id[i] specifies to which mesh element from the associated MeshBox the i-th rectangular region is mapped to. When the TileMeshGroupEntry is included in a track that also contains a MeshBox, the mesh_element_id[i] is associated with that MeshBox. When the TileMeshGroupEntry is included in an HEVC tile track which contains a 'tbas' track reference to an HEVC tile base track, the mesh_element_id[i] is associated with the MeshBox in that tile base track.

The pair of mesh_element_id[i] and eye[i] of the same loop entry shall not be equal to any pair of mesh_element_id[i] and eye[i] of any other loop entry among all the TileMeshGroupEntry sample group description entries in the same SampleGroupDescriptionBox.

RectRegionStruct() specifies a rectangular region. The regions described by any two instances of RectRegionStruct() in the same TileMeshGroupEntry shall not overlap.

GuardBand(i) specifies the guard bands around the i-th rectangular region. The syntax and semantics of GuardBand(i) are specified in subclauses 7.5.3.4 and 7.5.3.5, respectively.

NOTE Guard bands exclude the corners as illustrated in Figure 19.

7.16.3 Rectangular region structure

7.16.3.1 Definition

A `RectRegionStruct()` is used in `TileMeshGroupEntry` to specify a rectangular region that is mapped onto 3D mesh.

7.16.3.2 Syntax

```
aligned(8) class RectRegionStruct() {
   unsigned int(16) region_start_x;
   unsigned int(16) region_start_y;
   int(16) region_width;
   int(16) region_height;
}
```

7.16.3.3 Semantics

`region_start_x` and `region_start_y` specify the horizontal offset and the vertical offset of the start point, respectively, of a rectangular region in units of luma samples. `region_start_x` equal to 0 specifies the left-most luma sample column of the content decoded from the track containing this structure. `region_start_y` equal to 0 specifies the top-most luma sample row of the content decoded from the track containing this structure.

NOTE When a tile mesh sample group is present in an HEVC tile track, `region_start_x` and `region_start_y` are relative to the top-left corner of the tile(s) carried in any sample of the HEVC tile track rather than the top-left corner of the picture represented by any sample of the associated HEVC tile base track.

`region_width` and `region_height` specify the width and height of a rectangular region. If `region_width` is positive, the region continues to the right side of the start point, otherwise it continues to the left side. If the `region_height` is positive, the region continues below the start point, otherwise it continues above the start point. Figure 26 and Figure 27 illustrate examples of rectangular regions when `region_width` and `region_height` are positive and negative, respectively.

The sum of `region_start_x` and `region_width` shall be greater than or equal to 0.

The sum of `region_start_x` and `region_width` shall be less than the `width` of the `VisualSampleEntry`.

The sum of `region_start_y` and `region_height` shall be greater than or equal to 0.

The sum of `region_start_y` and `region_height` shall be less than the `height` of the `VisualSampleEntry`.

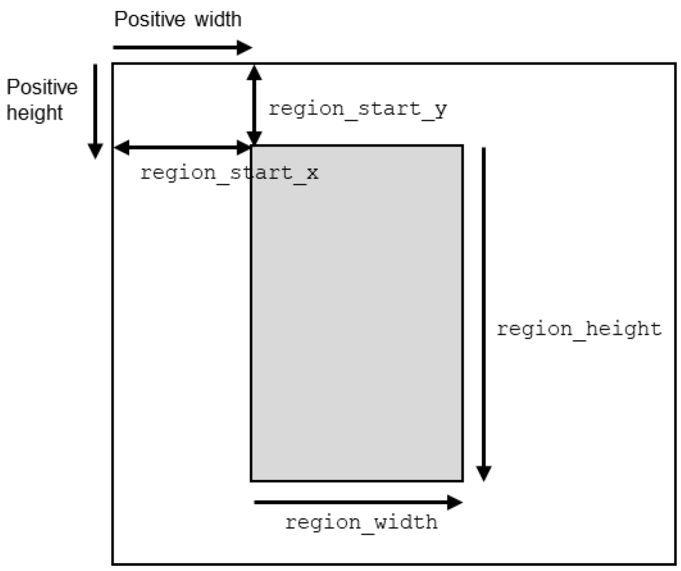


Figure 26 — Rectangular region when `region_width` and `region_height` are positive

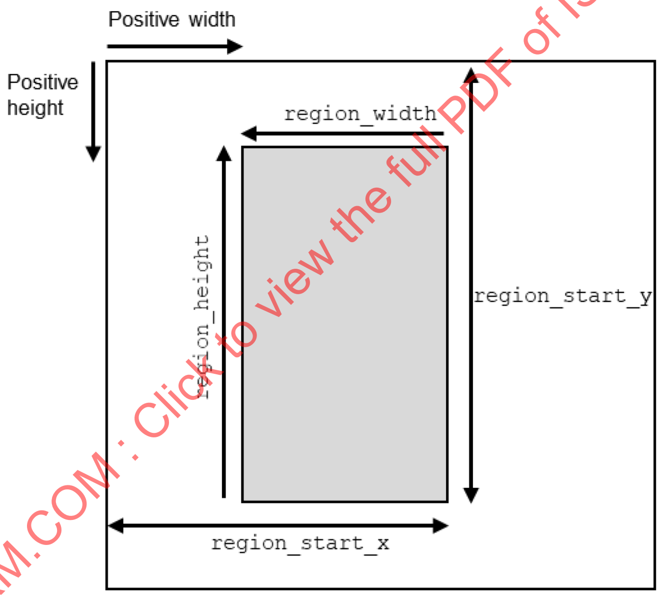


Figure 27 — Rectangular region when `region_width` and `region_height` are negative

7.16.4 Projection of a sample location onto the 3D mesh

In order to for an OMAF player to be able to use the `MeshBox` and the `TileMeshGroupEntry`, it needs to be able to map sample locations from a decoded picture to positions on the 3D mesh. This subclause specifies mapping of a sample location from a decoded picture to a position on the 3D mesh.

When a sample at coordinate (u,v) is inside the i-th `RectRegionStruct()`, the sample is projected onto a position P=(x,y,z) in the 3D mesh as follows:

— If `mesh_type` is equal to 0, the following applies:

  — The azimuth and elevation of point P are computed as follows:

$$\begin{aligned} pAzimuth &= ((u - \texttt{region\_start\_x} + 0.5) \div \texttt{region\_width}) * (cAzimuth2 - cAzimuth1) + cAzimuth1 \\ pElevation &= ((v - \texttt{region\_start\_y} + 0.5) \div \texttt{region\_height}) * (cElevation2 - cElevation1) + cElevation1 \end{aligned} \quad (19)$$

  where cAzimuth, cAzimuth2, cElevation1, and cElevation2 are specified in subclause 7.5.6.1.

  — The 3D position P=(x,y,z) is derived as follows:

$$\begin{aligned} x &= \cos(pAzimuth) * \cos(pElevation) \\ y &= \sin(pAzimuth) * \cos(pElevation) \\ z &= \sin(pElevation) \end{aligned} \quad (20)$$

— Otherwise (`mesh_type` is equal to 1), the 3D position P=(x,y,z) is derived as follows:

$$\mathbf{P} = \mathbf{origin\_vertex} \,\underline{+}\, \mathbf{u\_direction} \,\underline{*}\, ((u - \texttt{region\_start\_x}) \div (\texttt{region\_width} - 1)) \,\underline{+}\, \mathbf{v\_direction} \,\underline{*}\, ((v - \texttt{region\_start\_y}) \div (\texttt{region\_height} - 1)) \quad (21)$$

  with $\underline{+}$ being the vector addition where components of the first vector are added to the same components of the second vector, $\underline{*}$ being the scalar multiplication of a vector where each component of the vector is multiplied by the scalar, and bold variable names being 3D vectors.

— If `mesh_type` is equal to 2, the 3D position P=(x,y,z) is derived as follows:

  Apply equation (21) above with the following 3D vectors:

  — **origin_vertex** is a 3D vector with unit length pointing to the top left corner of the sphere region.

  — **u_direction** is a 3D vector from the top left corner to the bottom left corner

  — **v_direction** is a 3D vector from the top left corner to the top right corner

8 Omnidirectional media encapsulation and signalling in DASH

8.1 Architecture of DASH delivery in OMAF

Figure 28 illustrates the content flow in the DASH delivery function of OMAF.

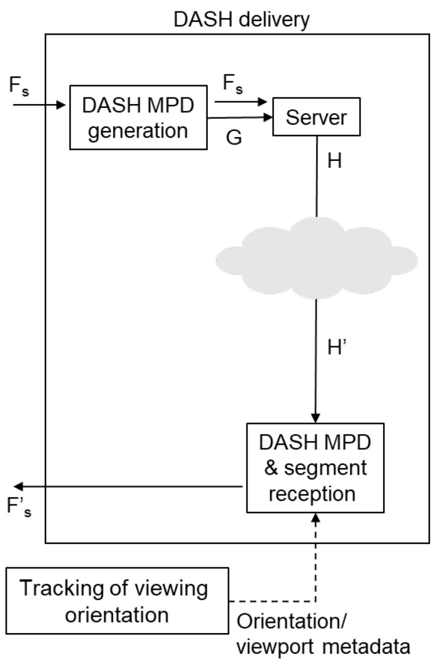

DASH delivery
F$_s$
DASH MPD generation
F$_s$
G
Server
H
H'
DASH MPD & segment reception
F'$_s$
Tracking of viewing orientation
Orientation/ viewport metadata

Figure 28 — Illustration of the content flow in the DASH delivery function of OMAF

The following interfaces are specified in this clause:

t) F$_s$/F'$_s$: initialization, index, and media segments; as defined generally below and specified for media profiles in Annex B. The OMAF DASH XML schema is provided in Annex A.

— G: DASH Media Presentation Description (MPD), including omnidirectional media-specific metadata, such as information on projection and region-wise packing, as specified in subclause 8.2.

An MPD (G) is generated based on the segments (F$_s$) and other media files representing the same content. The DASH MPD generator includes omnidirectional media-specific descriptors as specified in subclause 8.2. The descriptors include projection type, region-wise packing type, content coverage, spherical region-wise quality ranking, 2D region-wise quality ranking, and fisheye omnidirectional video information. These information may be generated on the basis of the equivalent information in the segments.

A DASH client obtains a current viewing orientation or viewport e.g. from the head-mounted display that detects the head and possibly also eye orientation. By parsing metadata, e.g. on projection and region-wise packing, from the MPD, the DASH client concludes which Adaptation Set and Representation contain the supported projection format etc. and which Adaptation Set and Representation cover the current viewing orientation at the highest quality and at a bitrate that may be afforded by the prevailing estimated network throughput. The DASH client issues (Sub)Segment requests accordingly.

The server typically provides segments (F$_s$) to the client. The server may also provide the MPD (considered as part of interface H in this case), or the MPD may be delivered by other means to the client. The segments and MPD are delivered over a network, and the received segments and MPD from the server are marked with H' in Figure 28. The output from the server (H) is considered to be identical to the input to the DASH MPD and segment reception block (H'). The received segments (F'$_s$) are output by the DASH MPD and segment reception block to the File/segment decapsulation block (see subclauses 4.2 and 4.4). In some media profiles, the segment reception function may include reconstruction of a conforming Segment sequence, which is regarded as the F'$_s$ interface.

8.2 Usage of DASH in OMAF

8.2.1 General

The following applies for the specifications in subclauses 8.2, 8.3, 8.4 and 8.5:

— The presence of an element at MPD level refers to that the element is a child element of the **`MPD`** element.

— The presence of an element at adaptation set level refers to that the element is a child element of an **`AdaptationSet`** element.

— The presence of an element at representation level refers to that the element is a child element of a **`Representation`** element.

8.2.2 Signalling of stereoscopic frame packing

A DASH **`FramePacking`** element with a `@schemeIdUri` attribute equal to `urn:mpeg:mpegB:cicp:VideoFramePackingType` may be present at adaptation set level and shall not be present at MPD or representation level. When used with projected omnidirectional video (i.e. when the PF descriptor is present), this element indicates that the projected pictures consist of spatially or temporally packed constituent pictures of the left and right views. The `@value` of the **`FramePacking`** element specifies the frame packing type for the stereoscopic video. This value shall be equal to 3, 4, or 5 with the meaning of those values as specified for `VideoFramePackingType` in ISO/IEC 23091-2. The value of `QuincunxSamplingFlag` as specified in ISO/IEC 23091-2 is inferred to be equal to 0.

8.2.3 Carriage of timed metadata

8.2.3.1 General

A timed metadata track, e.g. of track sample entry type `'invo'` as specified in subclause 7.7.4, `'rcvp'` as specified in subclause 7.7.5, or `'dyol'` as specified in subclause 7.14.6, may be encapsulated in a Representation.

8.2.3.2 Carriage of ERP region timed metadata in DASH

8.2.3.2.1 Constraints

ERP region timed metadata tracks specified in subclause 7.11 shall be carried in a DASH Adaptation Set with a single Representation with the constraint that the `@codecs` attribute shall have the value `'stmd'`.

When an ERP region timed metadata track is carried as a DASH Representation, the (Sub)Segment durations of ERP region timed metadata shall be aligned with the respective (Sub)Segments durations of the Representations carrying the associated sub-picture tracks. However, ERP region timed metadata (Sub)Segments may have a different number of samples than the sub-picture track (Sub)Segments.

NOTE A Representation carrying an ERP region timed metadata track can be retrieved faster than the actual media data representations to enable look-ahead.

8.2.3.2.2 OMAF player behaviour to process ERP region metadata

OMAF players are able to understand the presence of ERP region timed-metadata information in DASH MPD by checking the Adaptation Sets with single Representations with `@codecs` attributes set to `'stmd'`. OMAF players can further check the presence of `@associationId` attribute in order to associate the ERP region timed metadata with a viewpoint.

OMAF players can pre-fetch ERP region timed metadata tracks ahead of presentation time in order to interpret the heatmap and recommended quality and priority ranking information. They can then utilize this information in case of bandwidth fluctuations for sub-picture quality adaptation in order to maximize the viewer's visual experience and utilize the available bandwidth efficiently.

8.2.4 Associating Adaptation Sets or Representations with each other

8.2.4.1 Individual association

A timed metadata Representation may be associated with one or more media representations through individual association.

When individual association is in use, the following applies:

— The `@associationId` attribute of a timed metadata representation shall contain one or more values of the `@id` attribute of the representation(s) containing the omnidirectional media carried by the media track(s) that are associated with the timed metadata track through a `'cdsc'` track reference as specified in ISO/IEC 14496-12.

— The `@associationType` attribute of this metadata representation shall be equal to `'cdsc'`.

When present, a timed metadata Representation carrying a timed metadata track included in a `'vipo'` entity group shall be associated with the Representations carrying media track(s) included in the same `'vipo'` entity group through the `@associationId` attribute.

8.2.4.2 Collective association

A timed metadata representation may also be associated with one or more media representations through collective association.

A timed metadata representation may be collectively associated with all media representations of a sub-picture composition as follows: An association descriptor specified in subclause 8.5.2 is present as a child element of the DASH **Representation** element of the timed metadata representation, and the association descriptor shall:

— Include the following string in the **Association** element of the type: `"//AdaptationSet[omaf2:SubPicCompositionId="aa"]"`, where `"aa"` indicates the sub-picture composition identifier value.

— Include `'cdtg'` as the value of the **Association**`@associationKindList` attribute of the **Association** element.

A timed metadata representation may be collectively associated with all media representations of a viewpoint as follows: An association descriptor is present as a child element of the DASH **Representation** element of the timed metadata representation, and the association descriptor shall:

— Include the following string in the **Association** element: `"//AdaptationSet/Viewpoint[@schemeIdUri="urn:mpeg:mpegI:omaf:2018:vwpt" and @value="bb"]/.."`

where "`bb`" indicates the viewpoint ID value of the viewpoint as a string.

— Include `'cdtg'` as the value of the **Association**`@associationKindList` attribute of the **Association** element.

8.2.4.3 Overlay DASH association

When an Adaptation Set containing an overlay is associated with one or more Adaptation Sets containing background media, an association descriptor specified in subclause 8.5.2 shall be present as a child element of the **AdaptationSet** element containing the overlay.

In this case the association descriptor shall include both of the following:

— An XPath string in the **Association** element which evaluates to one or more **AdaptationSet** element(s) containing background media.

— Only one 'ovbg' value for the **Association**@associationKindList attribute of the **Association** element. In this case:

  — When **Association**@associationKindList includes one 'ovbg' value and the number of element(s) the XPath string in the **Association** element above evaluates to is greater than 1, the overlay applies collectively to the background media (e.g. if the background media is signalled via multiple Adaptation Sets with each Adaptation Set corresponding to a sub-picture).

  — When **Association**@associationKindList includes one 'ovbg' value and the number of elements the XPath string in the **Association** element above evaluates to is equal to 1, the overlay applies individually to the background media.

There can be multiple such association descriptors present inside an Adaptation Set containing an overlay. When an Adaptation Set containing an overlay is associated with one or more Adaptation Set(s) containing background media as described above, they are intended to be presented together.

8.3 DASH MPD descriptors for omnidirectional media in the namespace "urn:mpeg:mpegI:omaf:2017"

8.3.1 XML namespace and schema

A number of XML elements and attributes are defined in subclause 8.3 and its sublcuases. These XML elements are defined in a separate namespace "urn:mpeg:mpegI:omaf:2017". The namespace designator "omaf:" is used to refer to this name space in this document. These are specified in the schema documents in each subclause where a new MPD descriptor is specified. The namespace designator "xs:" shall correspond to namespace http://www.w3.org/2001/XMLSchema as defined in W3C Recommendation, *XML Schema Part 1: Structures*. Items in the "Data type" column of tables in subclause 8.2 use datatypes defined in W3C Recommendation, *XML Schema Part 2: Datatypes* and shall have the meaning as defined in W3C Recommendation, *XML Schema Part 2: Datatypes*.

8.3.2 Signalling of projection type information

An **EssentialProperty** element with a @schemeIdUri attribute equal to "urn:mpeg:mpegI:omaf:2017:pf" is referred to as a projection format (PF) descriptor.

At most one PF descriptor may be present at MPD level. At most one PF descriptor may be present at adaptation set level. At most one PF descriptor may be present at representation level.

The omaf:@projection_type attribute of a PF descriptor present at a hierarchically lower level overrides omaf:@projection_type attribute of a PF descriptor present at a hierarchically higher level. For example, when both an **AdaptationSet** element and a **Representation** element in the **AdaptationSet** element have a PF descriptor present, the PF descriptor present in the **Representation** element applies to the Representation.

The @value attribute of the PF descriptor shall not be present. The PF descriptor shall include an omaf:@projection_type attribute whose value shall not be empty as specified in Table 19.

Table 19 — Semantics of omaf:@projection_type attribute

Attribute for PF descriptor	Use	Data type	Description
omaf:@projection_type	M	omaf:listofUnsignedByte	Specifies a list of projection type values of the projected picture as specified in Table 10. Each value in the list shall be in the range of 0 to 31, inclusive. The values in the range of 32 to 255, inclusive, are reserved. Each value in the list shall be unique. For ISO Base Media File Format Segments, projection_type shall be equal to projection_type in ProjectionFormatBox in sample entries of the Initialization Segment.

The data type for the attribute shall be as defined in the XML schema. An XML schema for projection type signalling shall be as shown below. The schema shall be represented in an XML schema that has namespace urn:mpeg:mpegI:omaf:2017 and is specified as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:mpeg:mpegI:omaf:2017"
    xmlns:omaf="urn:mpeg:mpegI:omaf:2017"
    elementFormDefault="qualified">
    <xs:attribute name="projection_type" type="omaf:listOfUnsignedByte"/>
    <xs:simpleType name="listOfUnsignedByte">
     <xs:restriction>
        <xs:simpleType>
            <xs:list itemType="xs:unsignedByte"/>
        </xs:simpleType>
        <xs:minLength value="1"/>
     </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

8.3.3 Signalling of region-wise packing type

An **EssentialProperty** element with a @schemeIdUri attribute equal to "urn:mpeg:mpegI:omaf:2017:rwpk" is referred to as a region-wise packing (RWPK) descriptor.

An RWPK descriptor indicates the applied region-wise packing types in the representation(s) associated with the descriptor.

At most one RWPK descriptor may be present at MPD level. At most one RWPK descriptor may be present at adaptation set level. At most one RWPK descriptor may be present at representation level.

The omaf:@packing_type attribute of a RWPK descriptor present at a hierarchically lower level overrides omaf:@packing_type attribute of a RWPK descriptor present at a hierarchically higher level. For example, when both an **AdaptationSet** element and a **Representation** element in the **AdaptationSet** element have a RWPK descriptor present, the RWPK descriptor present in the **Representation** element applies to the Representation.

The @value of the RWPK descriptor shall not be present. The RWPK descriptor may include an omaf:@packing_type attribute as specified in Table 20.

Table 20 — Semantics of omaf:@packing_type attribute

Attribute for RWPK descriptor	Use	Data type	Description
omaf:@packing_type	O	omaf:OptionallistofUnsignedByte	Specifies a list of the packing type value of the picture as specified in Table 11. Each value in the list shall be in the range of 0 to 15, inclusive. The values in the range of 16 to 255, inclusive, are reserved. Each value in the list shall be unique. For ISO Base Media File Format Segments, packing_type shall be equal to packing_type in RegionWisePackingBox in sample entries of the Initialization Segment. When this omaf:@packing_type attribute is not present or does not include any value in a RWPK descriptor, omaf:@packing_type value is inferred to be equal to 0.

The absence of a RWPK descriptor indicates that no region-wise packing has been applied.

The data type for the attribute shall be as defined in the XML schema. An XML schema for region-wise packing type signalling shall be as shown below. The schema shall be represented in an XML schema that has namespace urn:mpeg:mpegI:omaf:2017 and is specified as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:mpeg:mpegI:omaf:2017"
    xmlns:omaf="urn:mpeg:mpegI:omaf:2017"
    elementFormDefault="qualified">
    <xs:attribute name="packing_type" type="omaf:OptionallistOfUnsignedByte"/>
    <xs:simpleType name="OptionallistOfUnsignedByte">
     <xs:restriction>
         <xs:simpleType>
             <xs:list itemType="xs:unsignedByte"/>
         </xs:simpleType>
         <xs:minLength value="0"/>
     </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

8.3.4 Signalling of content coverage

A **SupplementalProperty** element with a @schemeIdUri attribute equal to "urn:mpeg:mpegI:omaf:2017:cc" is referred to as a content coverage (CC) descriptor.

At most one CC descriptor may be present at adaptation set level. A CC descriptor shall not be present at MPD or representation level.

The CC descriptor indicates that each Representation covers the sphere region as specified in subclause 7.5.6 by shape_type and syntax elements centre_azimuth, centre_elevation, centre_tilt, azimuth_range, and elevation_range in SphereRegionStruct() as included in the CC descriptor.

When the CC descriptor that applies to a Representation is present and the CoverageInformationBox is present in the track corresponding to the Representation, the CC descriptor shall carry equivalent information as the CoverageInformationBox.

The @value attribute of the CC descriptor shall not be present. The CC descriptor shall include elements and attributes as specified in Table 21.

Table 21 — Semantics of elements and attributes of CC descriptor

Elements and attributes for CC descriptor	Use	Data type	Description
cc	0..1	omaf:CCType	Container element whose attributes and elements specify sphere region coverage information.
cc@shape_type	O	xs:unsignedByte	Specifies the shape type of the sphere region, as specified in 7.7.2.3. When not present, **cc**@shape_type is inferred to be equal to 0.
cc@view_idc_presence_flag	O	xs:boolean	Value 0 specifies that **cc.coverageInfo**@view_idc is not signalled. Value 1 specifies that **cc.coverageInfo**@view_idc is signalled and indicates the association of sphere regions with particular (left, right, or both) views or monoscopic content. When not present, **cc**@vview_idc_presence_flag is inferred to be equal to 0.
cc@default_view_idc	CM	omaf:ViewType	Value 0 indicates that all the sphere regions are monoscopic. Value 1 indicates that all the sphere regions are on the left view of a stereoscopic content. Value 2 indicates that all the sphere regions are on the right view of a stereoscopic content. Value 3 indicates that all the sphere regions are on both the left and right views. **cc**@default_view_idc shall be present when **cc**@view_idc_presence_flag is equal to 0. **cc**@default_view_idc shall be absent when **cc**@view_idc_presence_flag is equal to 1.
cc.coverageInfo	1..255	omaf:coverageInfoType	Element whose attribute **cc.coverageInfo**@view_idc, when present, provides information about view(s) to which coverage specified by sphere region defined by attributes **cc.coverageInfo**@centre_azimuth, **cc.coverageInfo**@centre_elevation, **cc.coverageInfo**@centre_tilt, **cc.coverageInfo**@azimuth_range, **cc.coverageInfo**@elevation_range applies.
cc.coverageInfo@view_idc	CM	omaf:ViewType	Value 1 indicates that the sphere region is on the left view of a stereoscopic content, value 2 indicates the sphere region is on the right view of a stereoscopic content, and value 3 indicates that the sphere region is on both the left and right views. Value 0 is reserved. **cc.coverageInfo**@view_idc shall be absent when **cc**@view_idc_presence_flag is equal to 0. **cc.coverageInfo**@view_idc shall be present when **cc**@view_idc_presence_flag is equal to 1.
cc.coverageInfo@centre_azimuth	O	omaf:Range1	Specifies the azimuth of the centre point of the sphere region in units of 2^{-16} degrees relative to the global coordinate axes. When not present, **cc.coverageInfo**@centre_azimuth is inferred to be equal to 0.
cc.coverageInfo@centre_elevation	O	omaf:Range2	Specifies the elevation of the centre point of the sphere region in units of 2^{-16} degrees relative to the global coordinate axes. When not present, **cc.coverageInfo**@centre_elevation is inferred to be equal to 0.
cc.coverageInfo@centre_tilt	O	omaf:Range1	Specifies the tilt angle of the sphere region, in units of 2^{-16} degrees, relative to the global coordinate axes. When not present, **cc.coverageInfo**@centre_tilt is inferred to be equal to 0.

Elements and attributes for CC descriptor	Use	Data type	Description
cc.coverageInfo@azimuth_range	O	omaf:HRange	Specifies the azimuth range of the sphere region through the centre point of the sphere region in units of 2^{-16} degrees. When not present **cc.coverageInfo**@azimuth_range is inferred to be equal to 360 * 2^{16}.
cc.coverageInfo@elevation_range	O	omaf:VRange	Specifies the elevation range of the sphere region through the centre point of the sphere region in units of 2^{-16} degrees. When not present **cc.coverageInfo**@elevation_range is inferred to be equal to 180 * 2^{16}.

NOTE When this descriptor is used with projected omnidirectional video, this descriptor can be used to indicate whether the sub-picture track carried in the Representation is monoscopic or stereoscopic by the attributes **cc**@default_view_idc or **cc.coverageInfo**@view_idc.

The absence of the **cc** element in the CC descriptor indicates that each Representation covers the entire sphere when a PF descriptor that applies to the Representation is present.

When a PF descriptor is not present at MPD level or adaptation set level, there shall be no CC descriptor present in the **AdaptationSet** element.

The data types for various elements and attributes shall be as defined in the XML schema. An XML schema for the CC descriptor shall be as shown below. The schema shall be represented in an XML schema that has namespace urn:mpeg:mpegI:omaf:2017 and is specified as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:mpeg:mpegI:omaf:2017"
    xmlns:omaf="urn:mpeg:mpegI:omaf:2017"
    elementFormDefault="qualified">
    <xs:element name="cc" type="omaf:CCType"/>
    <xs:complexType name="CCType">
       <xs:sequence>
            <xs:element name="coverageInfo" type="omaf:coverageInfoType"
            minOccurs="1" maxOccurs="255"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
            maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="shape_type" type="xs:unsignedByte" use="optional"
default="0"/>
     <xs:attribute name="view_idc_presence_flag" type="xs:boolean"
use="optional" default="0"/>
        <xs:attribute name="default_view_idc" type="omaf:ViewType"
use="optional"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
   </xs:complexType>
    <xs:complexType name="coverageInfoType">
        <xs:attribute name="view_idc" type="omaf:ViewType" use="optional"/>
        <xs:attribute name="centre_azimuth" type="omaf:Range1" use="optional"
default="0"/>
        <xs:attribute name="centre_elevation" type="omaf:Range2" use="optional"
default="0"/>
        <xs:attribute name="centre_tilt" type="omaf:Range1" use="optional"
default="0"/>
        <xs:attribute name="azimuth_range" type="omaf:HRange" use="optional"
default="23592960"/>
        <xs:attribute name="elevation_range" type="omaf:VRange" use="optional"
default="11796480"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:simpleType name="Range1">
```

```
            <xs:restriction base="xs:int">
                <xs:minInclusive value="-11796480"/>
                <xs:maxInclusive value="11796479"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="Range2">
            <xs:restriction base="xs:int">
                <xs:minInclusive value="-5898240"/>
                <xs:maxInclusive value="5898240"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="HRange">
            <xs:restriction base="xs:unsignedInt">
                <xs:minInclusive value="0"/>
                <xs:maxInclusive value="23592960"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="VRange">
            <xs:restriction base="xs:unsignedInt">
                <xs:minInclusive value="0"/>
                <xs:maxInclusive value="11796480"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="ViewType">
            <xs:restriction base="xs:unsignedByte">
                <xs:minInclusive value="0"/>
                <xs:maxInclusive value="3"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:schema>
```

8.3.5 Signalling of spherical region-wise quality ranking

A **`SupplementalProperty`** element with a `@schemeIdUri` attribute equal to `"urn:mpeg:mpegI:omaf:2017:srqr"` is referred to as a spherical region-wise quality ranking (SRQR) descriptor.

At most one SRQR descriptor for each **`sphRegionQuality`**`@shape_type` value of 0 and 1 may be present at adaptation set level. At most one SRQR descriptor for each **`sphRegionQuality`**`@shape_type` value of 0 and 1 may be present at representation level. A SRQR descriptor shall not be present at MPD level.

The SRQR descriptor should be present for Adaptation Sets or Representations containing projected omnidirectional video to enable viewport-dependent content selection.

The SRQR descriptor indicates a quality ranking value of a quality ranking sphere region relative to:

— other quality ranking sphere regions in the same Adaptation Set, and

— when **`sphRegionQuality`**`@quality_ranking_local_flag` is equal to 0, SRQR descriptors with **`sphRegionQuality`**`@quality_ranking_local_flag` equal to 0 and `@qualityRanking` values in all Adaptation Sets that have the same `@value` in the DASH **`Viewpoint`** element as the Adaptation Set containing this SRQR descriptor or containing the Representation that contains this SRQR descriptor.

NOTE 1 ISO/IEC 23009-1 specifies that the handling of the **`Viewpoint`** element is recommended to be applied equally for recognized and unrecognized `@schemeIdUri` values. It is suggested to use the same `@schemeIdUri` value for all **`Viewpoint`** elements in all Adaptation Sets of the same omnidirectional audio-visual content within the same Period.

The sphere region for the quality-ranking is specified by syntax elements `shape_type`, `centre_azimuth`, `centre_elevation`, `centre_tilt`, `azimuth_range`, `elevation_range` in `SphereRegionStruct()` as specified in subclause 7.5.6. When the quality ranking value **sphRegionQuality.qualityInfo**@quality_ranking is non-zero, the picture quality within the entire indicated quality ranking sphere region is approximately constant.

When the SRQR descriptor that applies to a Representation is present and a `SphereRegionQualityRankingBox` with the same shape type as that in the SRQR descriptor is present in the track corresponding to the Representation, the SRQR descriptor shall carry equivalent information as the `SphereRegionQualityRankingBox`.

The `@value` attribute of the SRQR descriptor shall not be present. The SRQR descriptor shall include a **sphRegionQuality** element with its sub-elements and attributes as specified in Table 22.

Table 22 — Semantics of elements and attributes of SRQR descriptor

Elements and attributes for SRQR descriptor	Use	Data type	Description
sphRegionQuality	1	omaf:SphRegionQualityType	Container element which includes one or more quality information elements (**sphRegionQuality.qualityInfo**) and common set of attributes (**sphRegionQuality**@shape_type, **sphRegionQuality**@remaining_area_flag, **sphRegionQuality**@view_idc_presence_flag, **sphRegionQuality**@quality_ranking_local_flag, **sphRegionQuality**@quality_Type, **sphRegionQuality**@default_view_idc) that apply to all those quality information elements.
sphRegionQuality@shape_type	0	xs:unsignedByte	Value 0 specifies that the quality ranking sphere region is indicated through four great circles as specified in subclause 7.7.2.3. Value 1 specifies that the quality ranking sphere region is indicated through two azimuth and two elevation circles as specified in subclause 7.7.2.3. When not present **sphRegionQuality**@shape_type is inferred to be equal to 0.
sphRegionQuality@remaining_area_flag	0	xs:boolean	Value 0 specifies that all the quality ranking sphere regions are specified by the signalled **sphRegionQuality.qualityInfo** elements. Value 1 specifies that all except the last quality ranking sphere regions are specified by the signalled **sphRegionQuality.qualityInfo** elements, and the last remaining quality ranking sphere region is the sphere region within the content coverage, not covered by the union of the quality ranking sphere regions specified by the signalled **sphRegionQuality.qualityInfo** elements. When not present **sphRegionQuality**@remaining_area_flag is inferred to be equal to 0. The last remaining quality ranking sphere region may be on both the left and right views. NOTE 2 When **sphRegionQuality**@remaining_area_flag is equal to 1, the **qualityInfo** element is present for the last quality ranking sphere region but excludes @centre_azimuth, @centre_elevation, @centre_tilt, @azimuth_range, and @elevation_range attributes.
sphRegionQuality@view_idc_presence_flag	0	xs:boolean	Value 0 specifies that **sphRegionQuality.qualityInfo**@view_idc is not signalled in each **sphRegionQuality.qualityInfo** element. Value 1 specifies that **sphRegionQuality.qualityInfo**@view_idc is signalled and indicates the association of quality ranking sphere regions with particular (left or right or both) views or monoscopic content. When not present **sphRegionQuality**@view_idc_presence_flag is inferred to be equal to 0.

Elements and attributes for SRQR descriptor	Use	Data type	Description
sphRegionQuality@quality_ranking_local_flag	O	xs:boolean	Value 0 specifies that the quality ranking information provided in this instance of this descriptor is relative to the quality ranking information provided in all instances of this descriptor with **sphRegionQuality**@quality_ranking_local_flag equal to 0 and in all instances of @qualityRanking in all Adaptation Sets that have the same @value in the DASH **Viewpoint** element as this Adaptation Set. Value 1 specifies that the quality ranking information provided in this instance of this descriptor is relative to the quality ranking information provided in all instances of this descriptor in this Adaptation Set only. When not present, the value of **sphRegionQuality**@quality_ranking_local_flag is inferred to be equal to 0. NOTE 3 **sphRegionQuality**@quality_ranking_local_flag equal to 1 can be used in the Main Adaptation Set of a Preselection and **sphRegionQuality**@quality_ranking_local_flag equal to 0 can be used in Adaptation Sets that are not the Main Adaptation Sets of a Preselection.
sphRegionQuality@quality_type	M	omaf:QualityType	Indicates which factor causes the differences in the quality of packed regions on the picture. Value 0 specifies that all packed regions correspond to the same projected picture resolution. Value 1 specifies that at least one horRatio value, as derived in subclause 5.4.2, may differ from other horRatio values among all pairs of packed and projected regions of the picture or at least one verRatio value, as derived in subclause 5.4.2, may differ from other verRatio values among all pairs of packed and projected regions of the picture. Values greater 1 are reserved.
sphRegionQuality@default_view_idc	CM	omaf:ViewType	Value 0 indicates that all the quality ranking sphere regions are monoscopic. Value 1 indicates that all the quality ranking sphere regions are on the left view of stereoscopic content. Value 2 indicates that all the quality ranking sphere regions are on the right view of stereoscopic content. Value 3 indicates that all the quality ranking sphere regions are on both the left and right views. **sphRegionQuality**@default_view_idc shall be present when **sphRegionQuality**@view_idc_presence_flag is equal to 0. **sphRegionQuality**@default_view_idc shall be absent when **sphRegionQuality**@view_idc_presence_flag is equal to 1.
sphRegionQuality.qualityInfo	1..255	omaf:QualityInfoType	Element whose attribute **sphRegionQuality.qualityInfo**@quality_ranking provides quality ranking for one quality ranking sphere region described by its attributes **sphRegionQuality.qualityInfo**@view_idc, **sphRegionQuality.qualityInfo**@centre_azimuth, **sphRegionQuality.qualityInfo**@centre_elevation, **sphRegionQuality.qualityInfo**@centre_tilt, **sphRegionQuality.qualityInfo**@azimuth_range, **sphRegionQuality.qualityInfo**@elevation_range.

Elements and attributes for SRQR descriptor	Use	Data type	Description
sphRegionQuality.qualityInfo @quality_ranking	M	xs:unsignedByte	Specifies a quality ranking value of the quality ranking sphere region. **sphRegionQuality.qualityInfo**@quality_ranking equal to 0 indicates that the quality ranking is not defined. When quality ranking sphere region A has a non-zero **sphRegionQuality.qualityInfo**@quality_ranking value less than the **sphRegionQuality.qualityInfo**@quality_ranking value of quality ranking sphere region B, quality ranking sphere region A has a higher quality than quality ranking sphere region B. When quality ranking sphere region A partly or entirely overlaps with quality ranking sphere region B, **sphRegionQuality.qualityInfo**@quality_ranking of quality ranking sphere region A shall be equal to **sphRegionQuality.qualityInfo**@quality_ranking of quality ranking sphere region B.
sphRegionQuality.qualityInfo @view_idc	CM	omaf:ViewType	Value 0 indicates that the content is monoscopic, value 1 indicates that the quality ranking sphere region is on the left view of stereoscopic content, value 2 indicates that the quality ranking sphere region is on the right view of stereoscopic content, 3 indicates that the quality ranking sphere region is on both the left and right views. **sphRegionQuality.qualityInfo**@view_idc shall be present when **sphRegionQuality**@view_idc_presence_flag is equal to 1. **sphRegionQuality.qualityInfo**@view_idc shall be absent when **sphRegionQuality**@view_idc_presence_flag is equal to 0.
sphRegionQuality.qualityInfo @orig_width	CM	xs:unsignedShort	Indicates the width of such a monoscopic projected picture for which horRatio, as derived in subclause 5.4.2 for each of the packed regions that cover the quality ranking sphere region, is equal to 1. Shall not be present when **sphRegionQuality**@quality_type is not equal to 1. Shall be present when **sphRegionQuality**@quality_type is equal to 1.
sphRegionQuality.qualityInfo @orig_height	CM	xs:unsignedShort	Indicates the height of such a monoscopic projected picture for which verRatio, as derived in subclause 5.4.2 for each of the packed regions that cover the quality ranking sphere region, is equal to 1. Shall not be present when **sphRegionQuality**@quality_type is not equal to 1. Shall be present when **sphRegionQuality**@quality_type is equal to 1.
sphRegionQuality.qualityInfo @centre_azimuth	CM	omaf:Range1	Specifies the azimuth of the centre point of the quality ranking sphere region, in units of 2^{-16} degrees, relative to the global coordinate axes. **sphRegionQuality.qualityInfo**@centre_azimuth shall be present when **sphRegionQuality**@remaining_area_flag is equal to 0. **sphRegionQuality.qualityInfo**@centre_azimuth shall be absent in only one **sphRegionQuality.qualityInfo** element and shall be present in all the other **sphRegionQuality.qualityInfo** elements when **sphRegionQuality**@remaining_area_flag is equal to 1.
sphRegionQuality.qualityInfo @centre_elevation	CM	omaf:Range2	Specifies the pitch of the centre point of the quality ranking sphere region, in units of 2^{-16} degrees, relative to the global coordinate axes. **sphRegionQuality.qualityInfo**@centre_elevation shall be present when **sphRegionQuality**@remaining_area_flag is equal to 0. **sphRegionQuality.qualityInfo**@centre_elevation shall be absent in only one **sphRegionQuality.qualityInfo** element and shall be present in all the other **sphRegionQuality.qualityInfo** elements when **sphRegionQuality**@remaining_area_flag is equal to 1.

Elements and attributes for SRQR descriptor	Use	Data type	Description
sphRegionQuality.qualityInfo @centre_tilt	CM	omaf:Range1	Specifies the tilt angle for the quality ranking sphere region in units of 2^{-16} degrees. **sphRegionQuality.qualityInfo**@centre_tilt shall be present when **sphRegionQuality**@remaining_area_flag is equal to 0. **sphRegionQuality.qualityInfo**@centre_tilt shall be absent one **sphRegionQuality.qualityInfo** element and shall be present in all the other **sphRegionQuality.qualityInfo** elements when **sphRegionQuality**@remaining_area_flag is equal to 1.
sphRegionQuality.qualityInfo @azimuth_range	CM	omaf:HRange	Specifies the azimuth range of the quality ranking sphere region through its centre point in units of 2^{-16} degrees. **sphRegionQuality.qualityInfo**@azimuth_range shall be present when **sphRegionQuality**@remaining_area_flag is equal to 0. **sphRegionQuality.qualityInfo**@azimuth_range shall be absent in only one **sphRegionQuality.qualityInfo** element and shall be present in all the other **sphRegionQuality.qualityInfo** elements when **sphRegionQuality**@remaining_area_flag is equal to 1.
sphRegionQuality.qualityInfo @elevation_range	CM	omaf:VRange	Specifies the elevation range of the quality raking sphere region through its centre point in units of 2^{-16} degrees. **sphRegionQuality.qualityInfo**@elevation_range shall be present when **sphRegionQuality**@remaining_area_flag is equal to 0. **sphRegionQuality.qualityInfo**@elevation_range shall be absent in only one **sphRegionQuality.qualityInfo** element and shall be present in all the other **sphRegionQuality.qualityInfo** elements when **sphRegionQuality**@remaining_area_flag is equal to 1.

NOTE 4 A player is suggested to parse spherical region-wise quality ranking (SRQR) descriptors and select the Adaptation Sets and Representations that matches the user's viewing orientation in a manner that:

— The quality ranking value on the region covering the viewport is greater than 0 and less than that for other regions.

— The resolution of the region covering the viewport is suitable for the display. If **sphRegionQuality**@quality_type is equal to 1, **sphRegionQuality.qualityInfo**@orig_width and **sphRegionQuality.qualityInfo**@orig_height represent the width and height of the monoscopic projected picture from which the packed region covering the viewport has been extracted. Otherwise, width and height of VisualSampleEntry can be used to conclude the resolution on the viewport.

The data types for various elements and attributes shall be as defined in the XML schema. An XML schema for SRQR is defined as shown below. The schema shall be represented in an XML schema that has namespace urn:mpeg:mpegI:omaf:2017 and is specified as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:mpeg:mpegI:omaf:2017"
    xmlns:omaf="urn:mpeg:mpegI:omaf:2017"
    elementFormDefault="qualified">

    <xs:element name="sphRegionQuality" type="omaf:SphRegionQualityType"/>
    <xs:complexType name="SphRegionQualityType">
        <xs:sequence>
            <xs:element name="qualityInfo" type="omaf:QualityInfoType"
minOccurs="1" maxOccurs="255"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="shape_type" type="xs:unsignedByte" use="optional"
default="0"/>
        <xs:attribute name="remaining_area_flag" type="xs:boolean"
```

```
use="optional" default="0"/>
        <xs:attribute name="view_idc_presence_flag" type="xs:boolean"
use="optional" default="0"/>
        <xs:attribute name="quality_ranking_local_flag" type="xs:boolean"
       use="optional" default="0"/>
        <xs:attribute name="quality_type" type="omaf:QualityType"
use="required"/>
        <xs:attribute name="default_view_idc" type="omaf:ViewType"
use="optional"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <xs:complexType name="QualityInfoType">
               <xs:attribute name="quality_ranking" type="xs:unsignedByte"
use="required"/>
               <xs:attribute name="view_idc" type="omaf:ViewType"
use="optional"/>
               <xs:attribute name="orig_width" type="xs:unsignedShort"
use="optional"/>
               <xs:attribute name="orig_height" type="xs:unsignedShort"
use="optional"/>
               <xs:attribute name="centre_azimuth" type="omaf:Range1"
use="optional"/>
               <xs:attribute name="centre_elevation" type="omaf:Range2"
use="optional"/>
               <xs:attribute name="centre_tilt" type="omaf:Range1"
use="optional"/>
               <xs:attribute name="azimuth_range" type="omaf:HRange"
use="optional"/>
               <xs:attribute name="elevation_range" type="omaf:VRange"
use="optional"/>
               <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:simpleType name="Range1">
        <xs:restriction base="xs:int">
            <xs:minInclusive value="-11796480"/>
            <xs:maxInclusive value="11796479"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="Range2">
        <xs:restriction base="xs:int">
            <xs:minInclusive value="-5898240"/>
            <xs:maxInclusive value="5898240"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="HRange">
        <xs:restriction base="xs:unsignedInt">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="23592960"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="VRange">
        <xs:restriction base="xs:unsignedInt">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="11796480"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="QualityType">
        <xs:restriction base="xs:unsignedByte">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="15"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="ViewType">
        <xs:restriction base="xs:unsignedByte">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="3"/>
        </xs:restriction>
        </xs:simpleType>
    </xs:schema>
```

8.3.6 Signalling of 2D region-wise quality ranking

A **`SupplementalProperty`** element with a `@schemeIdUri` attribute equal to `"urn:mpeg:mpegI:omaf:2017:2dqr"` is referred to as a 2D region-wise quality ranking (2DQR) descriptor.

At most one 2DQR descriptor may be present at adaptation set level. At most one 2DQR descriptor may be present at representation level. A 2DQR descriptor shall not be present at MPD level.

A 2DQR descriptor should be present for an Adaptation Set or a Representation containing projected omnidirectional video to enable viewport-dependent content selection, when no RWPK descriptor and no SRQR descriptor applies to the same Adaptation Set or Representation, respectively.

The 2DQR descriptor indicates a quality ranking value of a quality ranking 2D region relative to

— other quality ranking 2D regions in the same Adaptation Set, and

— when **`twoDRegionQuality`**`@quality_ranking_local_flag` is equal to 0, 2DQR descriptors with **`twoDRegionQuality`**`@quality_ranking_local_flag` equal to 0 and `@qualityRanking` values in all Adaptation Sets that have the same `@value` in the DASH **`Viewpoint`** element as the Adaptation Set containing this 2DQR descriptor or containing the Representation that contains this 2DQR descriptor.

NOTE 1 ISO/IEC 23009-1 specifies that the handling of the **`Viewpoint`** element is recommended to be applied equally for recognized and unrecognized `@schemeIdUri` values. It is suggested to use the same `@schemeIdUri` value for all **`Viewpoint`** elements in all Adaptation Sets of the same omnidirectional audio-visual content within the same Period.

When the quality ranking value **`twoDRegionQuality.twoDqualityInfo`**`@quality_ranking` is non-zero, the picture quality within the entire indicated quality ranking 2D region is approximately constant.

When the 2DQR descriptor that applies to a Representation is present and a `2DRegionQualityRankingBox` is present in the track corresponding to the Representation, the 2DQR descriptor shall carry equivalent information as the `2DRegionQualityRankingBox`.

The `@value` attribute of the 2DQR descriptor shall not be present. The 2DQR descriptor shall include a **`twoDRegionQuality`** element with its sub-elements and attributes as specified in Table 23.

Table 23 — Semantics of elements and attributes of 2DQR descriptor

Elements and attributes for 2DQR descriptor	Use	Data type	Description
`twoDRegionQuality`	1	omaf:twoDRegionQualityType	Container element which includes one or more 2D region quality information elements (**`twoDRegionQuality.twoDqualityInfo`**) and common set of attributes (**`twoDRegionQuality`**`@remaining_area_flag`, **`twoDRegionQuality`**`@view_idc_presence_flag`, **`twoDRegionQuality`**`@quality_ranking_local_flag`, **`twoDRegionQuality`**`@quality_type`, **`twoDRegionQuality`**`@default_view_idc`) that apply to all those quality information elements.

Elements and attributes for 2DQR descriptor	Use	Data type	Description
twoDRegionQuality@remaining_area_flag	O	xs:boolean	Value 0 specifies that all the quality ranking 2D regions are specified by the signalled **twoDRegionQuality.twoDqualityInfo** elements. Value 1 specifies that all except the last quality ranking 2D regions are specified by the signalled **twoDRegionQuality.twoDqualityInfo** elements, and the last remaining quality ranking 2D region is the 2D region within the content coverage, not covered by the union of the quality ranking 2D regions specified by the signalled **twoDRegionQuality.twoDqualityInfo** elements. When not present **twoDRegionQuality**@remaining_area_flag is inferred to be equal to 0. The last remaining quality ranking 2D region may be on both the left and right views. NOTE 2 When **twoDRegionQuality**@remaining_area_flag is equal to 1, the **twoDqualityInfo** element is present for the last quality ranking 2D region but excludes @left_offset, @top_offset, @region_width, and @region_height attributes.
twoDRegionQuality@view_idc_presence_flag	O	xs:boolean	Value 0 specifies that **twoDRegionQuality.twoDqualityInfo**@view_idc is not signalled. Value 1 specifies that **twoDRegionQuality.twoDqualityInfo**@view_idc is signalled and indicates the association of quality ranking 2D regions with particular (left or right or both) views or monoscopic content. When not present **twoDRegionQuality**@view_idc_presence_flag is inferred to be equal to 0.
twoDRegionQuality@quality_ranking_local_flag	O	xs:boolean	Value 0 specifies that the quality ranking information provided in this instance of this descriptor is relative to the quality ranking information provided in all instances of this descriptor with **twoDRegionQuality**@quality_ranking_local_flag equal to 0 and in all instances of @qualityRanking in all Adaptation Sets that have the same @value in the DASH **Viewpoint** element as this Adaptation Set. Value 1 specifies that the quality ranking information provided in this instance of this descriptor is relative to the quality ranking information provided in all instances of this descriptor in this Adaptation Set only. When not present, the value of **twoDRegionQuality**@quality_ranking_local_flag is inferred to be equal to 0. NOTE 3 **twoDRegionQuality**@quality_ranking_local_flag equal to 1 can be used in the Main Adaptation Set of a Preselection and **twoDRegionQuality**@quality_ranking_local_flag equal to 0 can be used in Adaptation Sets that are not the Main Adaptation Sets of a Preselection.
twoDRegionQuality@quality_type	M	omaf:QualityType	Indicates which factor causes the differences in the quality of packed regions on the picture. Value 0 specifies that all packed regions correspond to the same projected picture resolution. Value 1 specifies that at least one horRatio value, as derived in subclause 5.4.2, may differ from other horRatio values among all pairs of packed and projected regions of the picture or at least one verRatio value, as derived in subclause 5.4.2, may differ from other verRatio values among all pairs of packed and projected regions of the picture. Values greater 1 are reserved.

Elements and attributes for 2DQR descriptor	Use	Data type	Description
twoDRegionQuality@default_view_idc	CM	omaf:ViewType	Value 0 indicates that all the quality ranking 2D regions are monoscopic. Value 1 indicates that all the quality ranking 2D regions are on the left view of stereoscopic content. Value 2 indicates that all the quality ranking 2D regions are on the right view of stereoscopic content. Value 3 indicates that all the quality ranking 2D regions are on both the left and right views. **twoDRegionQuality**@default_view_idc shall be present when **twoDRegionQuality**@view_idc_presence_flag is equal to 0. **twoDRegionQuality**@default_view_idc shall be absent when **twoDRegionQuality**@view_idc_presence_flag is equal to 1.
twoDRegionQuality.twoDqualityInfo	1..255	omaf:twoDQualityInfoType	Element whose attribute **twoDRegionQuality.twoDqualityInfo**@quality_ranking provides quality ranking for one quality ranking 2D region described by its attributes **twoDRegionQuality.twoDqualityInfo**@view_idc, **twoDRegionQuality.twoDqualityInfo**@left_offset, **twoDRegionQuality.twoDqualityInfo**@top_offset, **twoDRegionQuality.twoDqualityInfo**@region_width, **twoDRegionQuality.twoDqualityInfo**@region_height.
twoDRegionQuality.twoDqualityInfo@quality_ranking	M	xs:unsignedByte	Specifies a quality ranking value of the quality ranking 2D region. **twoDRegionQuality.twoDqualityInfo**@quality_ranking equal to 0 indicates that the quality ranking is not defined. When quality ranking 2D region A has a non-zero **twoDRegionQuality.twoDqualityInfo**@quality_ranking value less than the **twoDRegionQuality.twoDqualityInfo**@quality_ranking value of quality ranking 2D region B, quality ranking 2D region A has a higher quality than quality ranking 2D region B. When quality ranking 2D region A partly or entirely overlaps with quality ranking 2D region B, **twoDRegionQuality.twoDqualityInfo**@quality_ranking of quality ranking 2D region A shall be equal to **twoDRegionQuality.twoDqualityInfo**@quality_ranking of quality ranking 2D region B.
twoDRegionQuality.twoDqualityInfo@view_idc	CM	omaf:ViewType	Value 0 indicates that the content is monoscopic, value 1 indicates that the quality ranking 2D region is on the left view of stereoscopic content, value 2 indicates that the quality ranking 2D region is on the right view of stereoscopic content, 3 indicates that the quality ranking 2D region is on both the left and right views. **twoDRegionQuality.twoDqualityInfo**@view_idc shall be present when **twoDRegionQuality**@view_idc_presence_flag is equal to 1. **twoDRegionQuality.twoDqualityInfo**@view_idc shall be absent when **twoDRegionQuality**@view_idc_presence_flag is equal to 0.
twoDRegionQuality.twoDqualityInfo@orig_width	CM	xs:unsignedShort	Indicates the width of such a monoscopic projected picture for which horRatio, as derived in subclause 5.4.2 for each of the packed regions that cover the quality ranking sphere region, is equal to 1. Shall not be present when **twoDRegionQuality**@quality_type is not equal to 1. Shall be present when **twoDRegionQuality**@quality_type is equal to 1.
twoDRegionQuality.twoDqualityInfo@orig_height	CM	xs:unsignedShort	Indicates the height of such a monoscopic projected picture for which verRatio, as derived in subclause 5.4.2 for each of the packed regions that cover the quality ranking sphere region, is equal to 1. Shall not be present when **twoDRegionQuality**@quality_type is not equal to 1. Shall be present when **twoDRegionQuality**@quality_type is equal to 1.

Elements and attributes for 2DQR descriptor	Use	Data type	Description
twoDRegionQuality.twoDqualityInfo@left_offset	CM	xs:unsignedShort	Specifies the horizontal coordinate of the upper left corner of the quality ranking 2D region within the picture in units of luma samples. **twoDRegionQuality.twoDqualityInfo**@left_offset shall be present when **twoDRegionQuality**@remaining_area_flag is equal to 0. **twoDRegionQuality.twoDqualityInfo**@left_offset shall be absent in only one **twoDRegionQuality.twoDqualityInfo** element and shall be present in all the other **twoDRegionQuality.twoDqualityInfo** elements when **twoDRegionQuality**@remaining_area_flag is equal to 1.
twoDRegionQuality.twoDqualityInfo@top_offset	CM	xs:unsignedShort	Specifies the vertical coordinate of the upper left corner of the quality ranking 2D region within the picture in units of luma samples. **twoDRegionQuality.twoDqualityInfo**@top_offset shall be present when **twoDRegionQuality**@remaining_area_flag is equal to 0. **twoDRegionQuality.twoDqualityInfo**@top_offset shall be absent in only one **twoDRegionQuality.twoDqualityInfo** element and shall be present in all the other **twoDRegionQuality.twoDqualityInfo** elements when **twoDRegionQuality**@remaining_area_flag is equal to 1.
twoDRegionQuality.twoDqualityInfo@region_width	CM	xs:unsignedShort	Specifies the width of the quality ranking 2D region within the picture in units of luma samples. **twoDRegionQuality.twoDqualityInfo**@region_width shall be present when **twoDRegionQuality**@remaining_area_flag is equal to 0. **twoDRegionQuality.twoDqualityInfo**@region_width shall be absent in only one **twoDRegionQuality.twoDqualityInfo** element and shall be present in all the other **twoDRegionQuality.twoDqualityInfo** elements when **twoDRegionQuality**@remaining_area_flag is equal to 1.
twoDRegionQuality.twoDqualityInfo@region_height	CM	xs:unsignedShort	Specifies the height of the quality ranking 2D region within the picture in units of luma samples. **twoDRegionQuality.twoDqualityInfo**@region_height shall be present when **twoDRegionQuality**@remaining_area_flag is equal to 0. **twoDRegionQuality.twoDqualityInfo**@region_height shall be absent in only one **twoDRegionQuality.twoDqualityInfo** element and shall be present in all the other **twoDRegionQuality.twoDqualityInfo** elements when **twoDRegionQuality**@remaining_area_flag is equal to 1.

NOTE 4 A player is suggested to parse 2D region-wise quality ranking (2DQR) descriptors and select the Adaptation Sets and Representations that matches the user's viewing orientation in a manner that:

— The quality ranking value on the region covering the viewport is greater than 0 and less than that for other regions.

— The resolution of the region covering the viewport is suitable for the display. If **twoDRegionQuality**@quality_type is equal to 1, **twoDRegionQuality.twoDqualityInfo**@orig_width and **twoDRegionQuality.twoDqualityInfo**@orig_height represent the width and height of the monoscopic projected picture from which the packed region covering the viewport has been extracted. Otherwise, `width` and `height` of `VisualSampleEntry` can be used to conclude the resolution on the viewport.

The data types for various elements and attributes shall be as defined in the XML schema. An XML schema for 2DQR is defined as shown below. The schema shall be represented in an XML schema that has namespace `urn:mpeg:mpegI:omaf:2017` and is specified as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:mpeg:mpegI:omaf:2017"
    xmlns:omaf="urn:mpeg:mpegI:omaf:2017"
    elementFormDefault="qualified">

    <xs:element name="twoDRegionQuality" type="omaf:twoDRegionQualityType"/>
    <xs:complexType name="twoDRegionQualityType">
        <xs:sequence>
            <xs:element name="twoDqualityInfo" type="omaf:twoDQualityInfoType"
minOccurs="1" maxOccurs="255"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="remaining_area_flag" type="xs:boolean"
use="optional" default="0"/>
        <xs:attribute name="view_idc_presence_flag" type="xs:boolean"
use="optional" default="0"/>
        <xs:attribute name="quality_ranking_local_flag" type="xs:boolean"
       use="optional"   default="0"/>
        <xs:attribute name="quality_type" type="omaf:QualityType"
use="required"/>
        <xs:attribute name="default_view_idc" type="omaf:ViewType"
use="optional"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <xs:complexType name="twoDQualityInfoType">
               <xs:attribute name="quality_ranking" type="xs:unsignedByte"
use="required"/>
               <xs:attribute name="view_idc" type="omaf:ViewType"
use="optional"/>
               <xs:attribute name="orig_width" type="xs:unsignedShort"
use="optional"/>
               <xs:attribute name="orig_height" type="xs:unsignedShort"
use="optional"/>
               <xs:attribute name="left_offset" type="xs:unsignedShort"
use="optional"/>
               <xs:attribute name="top_offset" type="xs:unsignedShort"
use="optional"/>
               <xs:attribute name="region_width" type="xs:unsignedShort"
use="optional"/>
               <xs:attribute name="region_height" type="xs:unsignedShort"
use="optional"/>
               <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
   <xs:simpleType name="QualityType">
        <xs:restriction base="xs:unsignedByte">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="15"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="ViewType">
        <xs:restriction base="xs:unsignedByte">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="3"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

8.3.7 Signalling of fisheye omnidirectional video

A **SupplementalProperty** element with a @schemeIdUri attribute equal to "urn:mpeg:mpegI:omaf:2017:fomv" is referred to as a fisheye omnidirectional video (FOMV) descriptor.

At most one FOMV descriptor may be present at adaptation set level. An FOMV descriptor shall not be present at MPD or representation level.

The FOMV descriptor indicates that each Representation carries a fisheye omnidirectional video track containing a FisheyeOmniVideoBox. The @value attribute of the FOMV descriptor shall not be present. The FOMV descriptor shall include an omaf:@view_dimension_idc attribute whose value shall be as specified in Table 24.

Table 24 — Semantics of omaf:@view_dimension_idc attribute

Attribute for FOMV descriptor	Use	Data type	Description
omaf:@view_dimension_idc	M	omaf:viewDIdcType	Has the same semantics as the view_dimension_idc syntax element (as specified in subclause 6.2.2) of the FisheyeVideoEssentialInfoStruct() syntax structure in the FisheyeOmniVideoBox in the tracks carried in the representations of this adaptation set. For ISO Base Media File Format Segments, view_dimension_idc shall be equal to view_dimension_idc in FisheyeVideoEssentialInfoBox in sample entries of the Initialization Segment.

The data type for the attribute shall be as defined in the XML schema. An XML schema for fisheye omnidirectional video signalling shall be as shown below. The schema shall be represented in an XML schema that has namespace urn:mpeg:mpegI:omaf:2017 and is specified as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:mpeg:mpegI:omaf:2017"
    xmlns:omaf="urn:mpeg:mpegI:omaf:2017"
    elementFormDefault="qualified">
    <xs:attribute name="view_dimension_idc" type="omaf:viewDIdcType"/>
    <xs:simpleType name="viewDIdcType">
        <xs:restriction base="xs:unsignedByte">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="7"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

8.4 Carriage of images

8.4.1 General

An Adaptation Set with media content component type described by the attribute @contentType="image" defines an Image Adaptation Set. An Image Adaptation Set contains alternate Representations, i.e. only one Representation within an Adaptation Set is expected to be presented at a time. Each Representation contained in one Image Adaptation Set contains one and only one image and is referred as an Image Representation. The Representations in an Image Adaptation Set are considered to be perceptually interchangeable.

The value of @mimeType attribute present or inferred for each Image Representation shall start with the type "image".

NOTE The subtype part of the value of @mimeType can take any values specified in the link http://www.iana.org/assignments/media-types/media-types.xhtml#image.

8.4.2 Format and constraints for Segments

An Image Representation shall consist of only one Self-Initializating Media Segment and shall not consist of any other Segments. Thus, the Media Segment itself conforms to the media type as specified in the @mimeType attribute for this Representation.

When the **SegmentBase** element is used for describing the Segment Information, **SegmentBase.Initialization** element shall not be present.

When the **SegmentList** element is used for describing the Segment Information, the **SegmentList.Initialization** element shall not be present.

When the **SegmentTemplate** element is used for describing the Segment Information, the **SegmentTemplate.Initialization** element and the **SegmentTemplate**@initialization attribute shall not be present.

8.5 DASH MPD descriptors for omnidirectional media in the namespace "urn:mpeg:mpegI:omaf:2020"

8.5.1 XML namespace and schema

A number of XML elements and attributes are defined in subclauses 8.5.2 to 8.5.6. These XML elements are defined in a separate namespace "urn:mpeg:mpegI:omaf:2020". These are specified in the schema documents in each subclause where a new MPD descriptor(s), element(s) or attribute(s) are specified. The namespace designator "xs:" shall correspond to namespace http://www.w3.org/2001/XMLSchema as defined in W3C Recommendation, *XML Schema Part 1: Structures*. Items in the "Data type" column of tables in subclauses 8.5.2 to 8.5.6 use datatypes defined in W3C Recommendation, *XML Schema Part 2: Datatypes* and shall have the meaning as defined in W3C Recommendation, *XML Schema Part 2: Datatypes*.

8.5.2 Signalling of association

A **SupplementalProperty** element with a @schemeIdUri attribute equal to "urn:mpeg:mpegI:omaf:2020:assoc" is referred to as an association descriptor.

One or more association descriptors may be present at adaptation set level, representation level, preselection level.

An association descriptor included inside an **AdaptationSet**, **Representation** or **Preselection** element indicates that the parent element of this element's descriptor (i.e. adaptation set/ representation/ preselection element) is associated with one or more elements in the MPD indicated by the XPath query in the omaf2:**Association** element and the association type signalled by omaf2:@associationKindList. The syntax of XPath strings used shall be as specified in W3C Recommendation, *XML Path Language (XPath) 2.0 (Second Edition)*.

The @value attribute of the association descriptor shall not be present. The association descriptor shall include one or more **Association** elements with attribute as specified in Table 25.

Table 25 — Semantics of elements and attributes of association descriptor

Elements and attributes for association descriptor	Use	Data type	Description
Association	0..N	omaf2:AssociationType	Element which specifies a list of XPath query string(s) which are evaluated to determine the elements (including certain values for their attributes) that are associated with the parent element of this **Association** element's descriptor. The XPath query shall evaluate to one or more elements.
Association@associationKindList	M	omaf2: listOfAssociationKindValues	Values in this list specify the kind of association between the parent element of this **Association** element's descriptor and the elements it is associated with. If this list includes a single entry then the parent element of this **Association** element's descriptor is associated collectively with all the elements resulting from evaluation of all XPath queries signalled in this **Association** element with the kind of association indicated by this attribute. If this list includes multiple entries then the number of entries in the list shall be equal to the number of entries in the list signalled in this attribute's **Association** element. In this case the parent element of this **Association** element's descriptor is associated with the element(s) specified by corresponding collocated XPath query in the **Association** element individually (if the XPath query results in a single element) or collectively if the XPath query results in multiple elements with the type of association indicated by the collocated value in this attribute.

The data types for various elements and attributes shall be as defined in the XML schema. An XML schema for this shall be as shown below. The schema shall be represented in an XML schema that has namespace `urn:mpeg:mpegI:omaf:2020` and is specified as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:mpeg:mpegI:omaf:2020"
    xmlns:omaf="urn:mpeg:mpegI:omaf:2017"
    xmlns:omaf2="urn:mpeg:mpegI:omaf:2020"
    elementFormDefault="qualified">
    <xs:element name="Association" type="omaf2:AssociationType"/>
    <xs:simpleType name="listOfAssociationValues">
        <xs:restriction>
            <xs:simpleType>
                <xs:list itemType="xs:string"/>
            </xs:simpleType>
            <xs:minLength value="1"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="FCCType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="listOfAssociationKindValues">
        <xs:restriction>
            <xs:simpleType>
                <xs:list itemType="omaf2:FCCType"/>
            </xs:simpleType>
            <xs:minLength value="1"/>
        </xs:restriction>
    </xs:simpleType>
```

```
        <xs:complexType name="AssociationType">
            <xs:simpleContent>
                <xs:extension base="omaf2:listOfAssociationValues">
                    <xs:attribute name="associationKindList"
    type="omaf2:listOfAssociationKindValues" use="required"/>
                    <xs:anyAttribute processContents="skip"/>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
    </xs:schema>
```

8.5.3 Signalling of viewpoints

In DASH MPD, a **`Viewpoint`** element with a `@schemeIdUri` attribute equal to `"urn:mpeg:mpegI:omaf:2020:vwpt"` is referred to as a viewpoint information (VWPT) descriptor.

At most one VWPT descriptor may be present at adaptation set level and no VWPT descriptor shall be present at any other level. When no Adaptation Set in the Media Presentation contains a VWPT descriptor, the Media Presentation is inferred to be contain only one viewpoint.

The VWPT descriptor indicates the viewpoint the Adaptation Set belongs to.

An Image Adaptation Set having a viewpoint information (VWPT) descriptor carries a viewpoint that is an image.

The VWPT descriptor shall include an `@value` attribute and a **`ViewPointInfo`** element with its sub-elements and attributes as specified in Table 26.

Table 26 — Semantics of elements and attributes of the VWPT descriptor

Elements and attributes for VWPT descriptor	Use	Data type	Description
`@value`	M	xs:string	Specifies the viewpoint ID of the viewpoint. The value is a string that contains a base-10 integer representation of a viewpoint ID that shall be equal to the `viewpoint_id` value of a viewpoint group entity grouping.
`ViewPointInfo`	1	omaf2:ViewPointInfoType	Container element whose sub-elements and attributes provide information about the viewpoint.
`ViewPointInfo` `@label`	0	xs:string	This attribute specifies a string that provides human readable label for the viewpoint.
`ViewPointInfo.Position`	1	omaf2:ViewpointPositionType	The attributes of this element specify the position information for the viewpoint.
`ViewPointInfo.Position``@x`	0	xs:int	Specifies the X position of the viewpoint, in units of 10^{-1} millimetres, in 3D space, relative to the common reference coordinate system. If position of the viewpoint is dynamic, this attribute specifies the initial viewpoint X position for this viewpoint. Otherwise, this attribute specifies the static viewpoint X position. When **`ViewPointInfo.Position`** is present but **`ViewPointInfo.Position`**`@x` is not present, **`ViewPointInfo.Position`**`@x` is inferred to be equal to zero.

Elements and attributes for VWPT descriptor	Use	Data type	Description
ViewPointInfo .Position@y	O	xs:int	Specifies the Y position of the viewpoint, in units of 10^{-1} millimetres, in 3D space, relative to the common reference coordinate system. If position of the viewpoint is dynamic, this attribute specifies the initial viewpoint Y position for this viewpoint. Otherwise, this attribute specifies the static viewpoint Y position. When **ViewPointInfo.Position** is present but **ViewPointInfo.Position**@y is not present, **ViewPointInfo.Position**@y is inferred to be equal to zero.
ViewPointInfo .Position@z	O	xs:int	Specifies the Z position of the viewpoint, in units of 10^{-1} millimetres, in 3D space, relative to the common reference coordinate system. If position of the viewpoint is dynamic, this attribute specifies the initial viewpoint Z position for this viewpoint. Otherwise, this attribute specifies the static viewpoint Z position. When **ViewPointInfo.Position** is present but **ViewPointInfo.Position**@z is not present, **ViewPointInfo.Position**@z is inferred to be equal to zero.
ViewPointInfo @initialViewpoint	O	xs:boolean	If equal to true this attribute specifies that this viewpoint is the initial viewpoint that should be used out of all the viewpoints in the current Period. If equal to false this attribute specifies that this viewpoint is not the initial viewpoint in the current Period. In a Period at most one viewpoint shall have **ViewPointInfo**@initialViewpoint equal to true. When no viewpoint in a Period has **ViewPointInfo**@initialViewpoint equal to true or if **ViewPointInfo**@initialViewpoint is not present then the initial viewpoint is specified by the associated initial viewpoint metadata representation. It should be avoided that a viewpoint is indicated as the initial viewpoint but not the main role.
ViewPointInfo .GpsPosition	0..1	omaf2:ViewpointGpsPositionType	The attributes of this element specify the GPS position information for the viewpoint.
ViewPointInfo .GpsPosition@longitude	M	omaf2:LongitudeRange	Indicates the longitude of the geolocation of the viewpoint in units of 2^{-23} degrees. The value shall be in range of $-180 * 2^{23}$ to $180 * 2^{23} - 1$, inclusive. Positive values represent eastern longitude and negative values represent western longitude. If position of the viewpoint is dynamic, this attribute specifies the initial viewpoint GPS position longitude of the geolocation for this viewpoint. Otherwise, this attribute specifies the static viewpoint GPS position longitude of the geolocation.
ViewPointInfo .GpsPosition@latitude	M	omaf2:LatitudeRange	Indicates the latitude of the geolocation of the viewpoint in units of 2^{-23} degrees. The value shall be in range of $-90 * 2^{23}$ to $90 * 2^{23} - 1$, inclusive. Positive value represents northern latitude and negative value represents southern latitude. If position of the viewpoint is dynamic, this attribute specifies the initial viewpoint GPS position latitude of the geolocation for this viewpoint. Otherwise, this attribute specifies the static viewpoint GPS position latitude of the geolocation.c

Elements and attributes for VWPT descriptor	Use	Data type	Description
ViewPointInfo.GpsPosition@altitude	M	xs:int	Indicates the altitude of the geolocation of the viewpoint in units of millimetres above the WGS 84 reference ellipsoid. NOTE The WGS 84 reference ellipsoid is specified in the EPSG:4326 database available at https://www.epsg.org/. If position of the viewpoint is dynamic, this attribute specifies the initial viewpoint GPS position altitude of the geolocation for this viewpoint. Otherwise, this attribute specifies the static viewpoint GPS position altitude of the geolocation.
ViewPointInfo.GeomagneticInfo	0..1	omaf2:ViewpointGeomagneticInfoType	The attributes of this element specify the relation between this viewpoint's common reference coordinate system relative to geomagnetic North direction. **ViewPointInfo.GeomagneticInfo** shall be present for at most one viewpoint in each viewpoint group.
ViewPointInfo.GeomagneticInfo@yaw	O	omaf:Range1	Specifies the yaw of the rotation angle of X, Y, Z axes of the common reference coordinate system relative to the geomagnetic North direction, in units of 2^{-16} degrees. The value shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive. When **ViewPointInfo.GeomagneticInfo** is present and **ViewPointInfo.GeomagneticInfo@**yaw is not present it is inferred to be equal to 0. If position of the viewpoint is dynamic, this attribute specifies the initial yaw of the rotation angle of the geolocation relative to geomagnetic North direction for this viewpoint. Otherwise, this attribute specifies the static viewpoint yaw rotation angle relative to geomagnetic North direction.
ViewPointInfo.GeomagneticInfo@pitch	O	omaf:Range2	Specifies the pitch of the rotation angle of X, Y, Z axes of the common reference coordinate system relative to the geomagnetic North direction, in units of 2^{-16} degrees.The value shall be in the range of $-90 * 2^{16}$ to $90 * 2^{16}$, inclusive. When **ViewPointInfo.GeomagneticInfo** is present and **ViewPointInfo.GeomagneticInfo@**pitch is not present it is inferred to be equal to 0. If position of the viewpoint is dynamic, this attribute specifies the initial pitch of the rotation angle of the geolocation relative to geomagnetic North direction for this viewpoint. Otherwise, this attribute specifies the static viewpoint pitch rotation angle relative to geomagnetic North direction.
ViewPointInfo.GeomagneticInfo@roll	O	omaf:Range1	Specifies the roll of the rotation angle of X, Y, Z axes of the common reference coordinate system relative to the geomagnetic North direction, in units of 2^{-16} degrees.The value shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive. When **ViewPointInfo.GeomagneticInfo** is present and **ViewPointInfo.GeomagneticInfo@**roll is not present it is inferred to be equal to 0. If position of the viewpoint is dynamic, this attribute specifies the initial roll of the rotation angle of the geolocation relative to geomagnetic North direction for this viewpoint. Otherwise, this attribute specifies the static viewpoint roll rotation angle relative to geomagnetic North direction.
ViewpointInfo.GroupInfo	1	omaf2:ViewpointGroupInfoType	The attributes of this element specify the viewpoint group information for the viewpoint. If viewpoint group information for a viewpoint is dynamic, this element and its attributes specify the initial viewpoint group information for this viewpoint. Otherwise, this element and its attributes specify the static viewpoint group information for this viewpoint.

Elements and attributes for VWPT descriptor	Use	Data type	Description
ViewPointInfo.GroupInfo@groupId	M	xs:unsignedByte	This attribute specifies the identifier of a viewpoint group that this viewpoint belongs to.
ViewpointInfo.GroupInfo@groupDescription	O	xs:string	This attribute specifies a string that provides a description of the viewpoint group identified by **ViewPointInfo.GroupInfo**@groupId. Absence of this attribute indicates that the viewpoint group does not have a description but is identified by the **ViewPointInfo.GroupInfo**@groupId attribute.
ViewpointInfo.SwitchingInfo	0..N	omaf2:ViewpointSwitchInfoType	The elements and attributes of this element specify the viewpoint switching list information
ViewpointInfo.SwitchingInfo.SwitchRegion	0..1	omaf2:ViewpointSwitchRegionType	The elements and attributes of this element specify the viewpoint switching region information
ViewpointInfo.SwitchingInfo.SwitchRegion@regionType	M	omaf2:restDatType	This attribute specifies the region type. Value equal to 0 specifies that the viewpoint switch region is positioned on the viewport. Value equal to 1 specifies that the viewpoint switch region is positioned on the sphere as a sphere region. Value equal to 2 specifies that the viewpoint switch region is specified by an overlay. Value equal to 3 is reserved.
ViewpointInfo.SwitchingInfo.SwitchRegion.VpRelative	0..1	omaf2:ViewpointRelativeType	This element and its attributes specify information about viewpoint switch region positioned on the viewport. ViewpointInfo.SwitchingInfo.VpRelative shall be present only when **ViewpointInfo.SwitchingInfo.switchRegion**@regionType is equal to 0.
ViewpointInfo.SwitchingInfo.SwitchRegion.VpRelative@rectLeftPct	M	xs:unsignedIShort	This attribute specifies the left corner of the viewpoint switch region, placed on the viewport in percent relative to the width of the viewport in units of 100% ÷ 2^{16} in the range of 0 (indicating 0%), inclusive, up to but excluding 65536 (that indicates 100%).
ViewpointInfo.SwitchingInfo.SwitchRegion.VpRelative@rectTopPct	M	xs:unsignedIShort	This attribute specifies the top corner of the viewpoint switch region, placed on the viewport in percent relative to the height of the viewport in units of 100% ÷ 2^{16} in the range of 0 (indicating 0%), inclusive, up to but excluding 65536 (that indicates 100%).
ViewpointInfo.SwitchingInfo.SwitchRegion.VpRelative@rectWidthPct	M	xs:unsignedIShort	This attribute specifies the width of the viewpoint switch region, placed on the viewport in percent relative to the width of the viewport in units of 100% ÷ 2^{16} in the range of 1 (indicating 0%), inclusive, up to but excluding 65536 (that indicates 100%).
ViewpointInfo.SwitchingInfo.SwitchRegion.VpRelative@rectHeightPct	M	xs:unsignedIShort	This attribute specifies the height of the viewpoint switch region, placed on the viewport in percent relative to the height of the viewport in units of 100% ÷ 2^{16} in the range of 1 (indicating 0%), inclusive, up to but excluding 65536 (that indicates 100%).

Elements and attributes for VWPT descriptor	Use	Data type	Description
ViewpointInfo.SwitchingInfo.SwitchRegion.SphereRegion	0..1	omaf2:SphereRegionStructType	This element and its attributes specify the specific viewport specified by sphere regions structure that shall be used after transitioning to the new viewpoint. **ViewpointInfo.SwitchingInfo.SwitchRegion.SphereRegion** shall be present only when **ViewpointInfo.SwitchingInfo.switchRegion**@regionType is equal to 1.
ViewpointInfo.SwitchingInfo.SwitchRegion.SphereRegion@centerAzimuth	M	omaf:Range1	Center point azimuth orientation
ViewpointInfo.SwitchingInfo.SwitchRegion.SphereRegion@centerElevation	M	omaf:Range2	Center point elevation orientation
ViewpointInfo.SwitchingInfo.SwitchRegion.SphereRegion@centerTilt	O	omaf:Range1	Tilt angle of the spatial playback region on the unit sphere. When absent, the value is '0'.
ViewpointInfo.SwitchingInfo.SwitchRegion.SphereRegion@azimuthRange	O	omaf:HRange	The azimuth range of the spatial playback region on the unit sphere. When absent, the value is the maximum value of the corresponding data type.
ViewpointInfo.SwitchingInfo.SwitchRegion.SphereRegion@elevationRange	O	omaf:VRange	The elevation range of the spatial playback region on the unit sphere. When absent, the value is the maximum value of the corresponding data type.
ViewpointInfo.SwitchingInfo.SwitchRegion.SphereRegion@shapeType	M	xs:unsignedByte	The geometric objects that define the region boundary "0" specifies that the region is specified by four great circles as shown in Figure 20. "1" specifies that the region is specified by two yaw circles and two pitch circles as illustrated in Figure 21. The specifics of calculating the circles are specified in subclause 7.5.6 **ViewpointInfo.SwitchingInfo.SwitchRegion.SphereRegion**@shapeType is used as the shape type value when applying subclause 7.5.6 to the semantics of the SphereRegionStruct(rangeIncluded, 0) specifying the viewpoint switch region where syntax element values of the SphereRegionStruct(rangeIncluded, 0) are populated by the respective syntax element values of **ViewpointInfo.SwitchingInfo.SwitchRegion.SphereRegion** and rangeIncluded is set equal to 0 if @azimuthRange and @elevationRange are absent and set equal to 1 otherwise.

Elements and attributes for VWPT descriptor	Use	Data type	Description
ViewpointInfo.SwitchingInfo.SwitchRegion@refOverlayId	O	xs:unsignedShort	When present, this attribute specifies the overlay_id of the overlay that specifies the viewpoint switch region. **ViewpointInfo.SwitchingInfo**@refOverlayId shall be present only when **ViewpointInfo.SwitchingInfo.switchRegion**@regionType is equal to 2.
ViewpointInfo.SwitchingInfo.SwitchRegion@id	M	xs:string	Identifier of the spatial playback region specified by **ViewpointInfo.SwitchingInfo.SwitchRegion**
ViewpointInfo.SwitchingInfo.SwitchRegion@region	O	xs:string	**ViewpointInfo.SwitchingInfo.SwitchRegion**@id from which the parent Adaptation Set in the Period whose identifier is **ViewpointInfo.SwitchingInfo.SwitchRegion**@period shall be selected as to be played back after the current Period.
ViewpointInfo.SwitchingInfo.SwitchRegion@period	M	xs:string	@id of the Period to be played back next (may not be the next Period in the MPD)
ViewpointInfo.SwitchingInfo.SwitchRegion@label	O	xs:string	A label associated with the spatial playback region specified by **ViewpointInfo.SwitchingInfo.SwitchRegion**

When **ViewpointInfo.SwitchingInfo.switchRegion**@regionType is equal to 1, the value of interpolate for the sphere region structure specified by **ViewpointInfo.SwitchingInfo.SwitchRegion.SphereRegion** is inferred to be equal to 0.

If the viewpoint is associated with a timed metadata Representation carrying a timed metadata track with sample entry type 'dyvp', the position of the viewpoint is dynamic. Otherwise, the position of the viewpoint is static. In the former case, the dynamic position of the viewpoint is signalled in the associated timed metadata Representation carrying a timed metadata track with sample entry type 'dyvp'.

The data types for various elements and attributes shall be as defined in the XML schema. An XML schema for this shall be as shown below. The schema shall be represented in an XML schema that has namespace urn:mpeg:mpegI:omaf:2020 and is specified as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:mpeg:mpegI:omaf:2020"
    xmlns:omaf="urn:mpeg:mpegI:omaf:2017"
    xmlns:omaf2="urn:mpeg:mpegI:omaf:2020"
    elementFormDefault="qualified">
    <xs:import namespace="urn:mpeg:mpegI:omaf:2017"
        schemaLocation="OMAFV1.xsd"/>
    <xs:element name="ViewpointInfo" type="omaf2:ViewpointInfoType"/>
    <xs:complexType name="ViewpointInfoType">
        <xs:sequence>
            <xs:element name="Position" type="omaf2:ViewpointPositionType"
minOccurs="1" maxOccurs="1"/>
            <xs:element name="GpsPosition"
type="omaf2:ViewpointGpsPositionType" minOccurs="0" maxOccurs="1"/>
           <xs:element name="GeomagneticInfo"
type="omaf2:ViewpointGeomagneticInfoType" minOccurs="0" maxOccurs="1"/>
```

```
            <xs:element name="GroupInfo" type="omaf2:ViewpointGroupInfoType"
minOccurs="1" maxOccurs="1"/>
            <xs:element name="SwitchingInfo"
type="omaf2:ViewpointSwitchInfoType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
            <xs:attribute name="label" type="xs:string" use="optional"/>
            <xs:attribute name="initialViewpoint" type="xs:boolean"
use="optional"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
    <xs:complexType name="ViewpointPositionType">
        <xs:attribute name="x" type="xs:int" use="optional" default="0"/>
        <xs:attribute name="y" type="xs:int" use="optional" default="0"/>
        <xs:attribute name="z" type="xs:int" use="optional" default="0"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="ViewpointGpsPositionType">
        <xs:attribute name="longitude" type="omaf2:LongitudeRange"
use="required"/>
        <xs:attribute name="latitude" type="omaf2:LatitudeRange"
use="required"/>
        <xs:attribute name="altitude" type="xs:int" use="required"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="ViewpointGeomagneticInfoType">
        <xs:attribute name="yaw" type="omaf:Range1" use="optional"
default="0"/>
        <xs:attribute name="pitch" type="omaf:Range2" use="optional"
default="0"/>
        <xs:attribute name="roll" type="omaf:Range1" use="optional"
default="0"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="ViewpointGroupInfoType">
        <xs:attribute name="groupId" type="xs:unsignedByte" use="required" />
        <xs:attribute name="groupDescription" type="xs:string" use="optional"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:simpleType name="LatitudeRange">
        <xs:restriction base="xs:int">
            <xs:minInclusive value="-754974720"/>
            <xs:maxInclusive value="754974719"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="LongitudeRange">
        <xs:restriction base="xs:int">
            <xs:minInclusive value="-1509949440"/>
            <xs:maxInclusive value="1509949439"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="ViewpointSwitchInfoType">
        <xs:sequence>
            <xs:element name="SwitchRegion"
type="omaf2:ViewpointSwitchRegionType" minOccurs="0" maxOccurs="1"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
                <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
    <xs:simpleType name="restDatType">
        <xs:restriction base="xs:byte">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="3"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="ViewpointSwitchRegionType">
```

```
        <xs:sequence>
            <xs:element name="VpRelative" type="omaf2:ViewpointRelativeType"
minOccurs="0" maxOccurs="1"/>
            <xs:element name="SphereRegion" type="omaf2:SphereRegionStructType"
minOccurs="0" maxOccurs="1"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="regionType" type="omaf2:restDatType"
use="required"/>
        <xs:attribute name="refOverlayId" type="xs:unsignedShort"
use="optional"/>
        <xs:attribute name="id" type="xs:string" use="required"/>
        <xs:attribute name="region" type="xs:string" use="optional"/>
        <xs:attribute name="period" type="xs:string" use="required"/>
        <xs:attribute name="label" type="xs:string" use="optional"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
    <xs:complexType name="ViewpointRelativeType">
        <xs:attribute name="rectLeftPct" type="xs:unsignedShort"
use="required"/>
        <xs:attribute name="rectTopPct" type="xs:unsignedShort"
use="required"/>
        <xs:attribute name="rectWidthPct" type="xs:unsignedShort"
use="required"/>
        <xs:attribute name="rectHeightPct" type="xs:unsignedShort"
use="required"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="SphereRegionStructType">
        <xs:attribute name="centreAzimuth" type="omaf:Range1" use="required"/>
        <xs:attribute name="centreElevation" type="omaf:Range2"
use="required"/>
        <xs:attribute name="centreTilt" type="omaf:Range1" use="required"/>
        <xs:attribute name="azimuthRange" type="omaf:HRange" use="optional"/>
        <xs:attribute name="elevationRange" type="omaf:VRange" use="optional"/>
        <xs:attribute name="shapeType" type="xs:unsignedByte" use="required"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
</xs:schema>
```

8.5.4 Signalling of sub-picture composition identifier and its attributes

Sub-picture representations carrying sub-picture tracks belonging to the same 2D spatial relationship track group may be indicated by a sub-picture composition identifier element **`SubPicCompositionId`** signalled as a child element of **`AdaptationSet`** element as specified in Table 27.

The **`SubPicCompositionId`** element may be present at adaptation set level and shall not be present at any other level.

An optional attribute `omaf2:@noSingleSelection` may be present as an attribute of **`SubPicCompositionId`** element.

Table 27 — Semantics of the SubPicCompositionId element and its attributes

Element	Use	Data type	Description
SubPicCompositionId	0..N	xs: unsignedShort	Specifies the identifier of an Adaptation Set that includes sub-picture representations carrying sub-picture tracks belonging to the same 2D spatial relationship track group. All Adaptation Sets in a Period that have the same value of `SubPicCompositionId` together form a sub-picture composition.
omaf2: **SubPicCompositionId**@noSingleSelection	O	xs:boolean	When omaf2:**SubPicCompositionId**@noSingleSelection is "true", it specifies that this Adaptation Set consists of a sub-picture and forms a part of a sub-picture composition identified by the `SubPicCompositionId` value of the parent element of this attribute and which is not intended to be selected alone for a presentation without at least one other Adaptation Set belonging to the same sub-picture composition identified by the `SubPicCompositionId` element that this attribute belongs to. When omaf2:**SubPicCompositionId**@noSingleSelection is "false", it specifies that this Adaptation Set may or may not be intended to be selected alone for a presentation without at least one other Adaptation Set belonging to the same sub-picture composition identified by the `SubPicCompositionId` element that this attribute belongs to. When omaf2:**SubPicCompositionId**@noSingleSelection is not present, it is inferred to be "false".

The data type for the element shall be as defined in the XML schema. An XML schema for this element shall be as shown below. The schema shall be represented in an XML schema that has namespace `urn:mpeg:mpegI:omaf:2020` and is specified as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="urn:mpeg:mpegI:omaf:2020"
        xmlns:omaf2="urn:mpeg:mpegI:omaf:2020"
        elementFormDefault="qualified">
  <xs:element name="SubPicCompositionId">
        <xs:complexType>
            <xs:simpleContent>
                <xs:extension base="xs:unsignedShort">
                    <xs:attribute name="noSingleSelection" type="xs:boolean"/>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
  </xs:element>
</xs:schema>
```

8.5.5 Signalling of overlays

An **EssentialProperty** or **SupplementalProperty** element with a `@schemeIdUri` attribute equal to `"urn:mpeg:mpegI:omaf:2020:ovly"` is referred to as an overlay information (OVLY) descriptor. An **EssentialProperty** element shall be used, when the displaying of the overlay is required, i.e. when the player is not allowed to turn the overlay off.

An OVLY descriptor indicates the overlays in the representation(s) associated with the descriptor.

At most one OVLY descriptor may be present at adaptation set level and no OVLY descriptor shall be present at any other level.

When a recommended viewport is used as an overlay, an OVLY descriptor shall be present with the Adaptation Set containing the referred recommended viewport timed metadata track.

An Image Adaptation Set having an overlay (OVLY) descriptor carries one or more overlay images.

The `@value` attribute of the OVLY descriptor shall not be present. The OVLY descriptor shall include attributes as specified in Table 28.

Table 28 — Semantics of the attributes of the OVLY descriptor

Elements and Attributes for OVLY descriptor	**Use**	**Data type**	**Description**
`@overlayIds`	M	omaf:listofUnsignedShort	Specifies a whitespace-separated list of overlay IDs of overlays as indicated by `overlay_id` as specified in subclause 7.14.2. Each specified overlay ID shall be unique.
`@priority`	O	omaf:listofUnsignedByte	Indicates a whitespace-separated list of overlay priorities of overlays as indicated by `overlay_priority` as specified in subclause 7.14.3.11. The number of overlay priorities shall be the same as the number of overlay IDs specified by the `@overlayIds` attribute, for the list of overlays in the same order as the list of overlay IDs specified by the `@overlayIds` attribute. A value of `@priority` equal to 255 indicates unspecified priority.

The data type for the element shall be as defined in the XML schema. An XML schema for this element shall be as shown below. The schema shall be represented in an XML schema that has namespace `urn:mpeg:mpegI:omaf:2020` and is specified as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:mpeg:mpegI:omaf:2020"
    xmlns:omaf="urn:mpeg:mpegI:omaf:2017"
    xmlns:omaf2="urn:mpeg:mpegI:omaf:2020"
    elementFormDefault="qualified">
    <xs:import namespace="urn:mpeg:mpegI:omaf:2017"
        schemaLocation="OMAFV1.xsd"/>
    <xs:simpleType name="listOfUnsignedShort">
        <xs:restriction>
            <xs:simpleType>
                <xs:list itemType="xs:unsignedShort"/>
            </xs:simpleType>
            <xs:minLength value="1"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="listOfUnsignedByte">
        <xs:restriction>
            <xs:simpleType>
                <xs:list itemType="xs:unsignedByte"/>
            </xs:simpleType>
            <xs:minLength value="1"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:attribute name="overlayIds" type="omaf2:listOfUnsignedShort"
use="required"/>
    <xs:attribute name="priority" type="omaf:listOfUnsignedByte" use="optional"/>
</xs:schema>
```

8.5.6 Entity to group descriptor

An **EssentialProperty** or **SupplementalProperty** element with a @schemeIdUri attribute equal to "urn:mpeg:mpegI:omaf:2020:etgb" is referred to as an EntityToGroup descriptor.

One or more EntityToGroup descriptors may be present at period level or at MPD level.

The @value attribute of the EntityToGroup descriptor shall not be present. The EntityToGroup descriptor shall include one or more **EntityGroup** elements with attributes and elements specified in Table 29. An **EntityGroup** element corresponds to an EntityToGroupBox contained in a GroupsListBox of a file-level MetaBox. **EntityGroup**@group_type carries a four-character code applicable as a grouping_type value of EntityToGroupBox.

EntityGroup@group_type equal to 'oval' specifies a set of Representations of overlays. Each overlay in the set is intended to be presented as a user-switchable alternative for another overlay in the same entity group.

Table 29 — Semantics of the elements and attributes of the EntityToGroup descriptor

Elements and Attributes for EntityToGroup descriptor	Use	Data type	Description
EntityGroup	1 ... N	omaf2: EntityGroupType Type	Container element which specifies an entity group. Its sub-element and attributes provide information about the entity group
EntityGroup@group_type	M	dash:FourCCType	This attribute is a four-character code that identifies the type (i.e. criterion used to form the entity groups) of the entity grouping. **EntityGroup**@group_type shall be equal to grouping_type of a single entity group contained in the Initialization Segment.
EntityGroup@group_id	M	xs:unsignedInt	This attribute is a non-negative 32-bit integer assigned to the entity group that shall not be equal to any **EntityGroup**@group_id value of any other **EntityGroup** element. **EntityGroup**@group_id shall be equal to group_id of the entity group with grouping_type equal to **EntityGroup**@group_type contained in the Initialization Segment.
EntityGroup.EntityIdList	1 ... N	omaf2:EntityIdType	The attributes of this element list all the Adaptation Sets and Representations which belong to this entity group
EntityGroup.EntityIdList@asid	M	xs:unsignedInt	Specifies the Adaptation Set ID that belong to the entity group

Elements and Attributes for EntityToGroup descriptor	Use	Data type	Description
EntityGroup.EntityIdList@rsid	M	dash:StringNoWhitespaceType	Specifies the Representation ID present in the Adaptation Set specified by the **EntityGroup.EntityIdList**@asid which belongs to the entity group
EntityGroup @ref_overlay_id	O	omaf2:listofUnsignedShort	This attribute is a whitespace-separated list of overlay IDs which are user-switchable alternative for another overlay in the same list. When **EntityGroup**@group_type is not equal to 'oval', **EntityGroup**@ref_overlay_id shall not be present. When present, the list **EntityGroup**@ref_overlay_id shall include at least one entry. The first entry in the list is the initial overlay that shall be rendered before any user interaction or switchable alternative selection. The whitespace-separated list of overlay IDs shall be identical to that specified by the list of ref_overlay_id[i] values in the 'oval' entity group with the group_id value equal to **EntityGroup**@group_id in the Initialization Segment.

The data types for various elements and attributes shall be as defined in the XML schema. An XML schema for this shall be as shown below. The schema shall be represented in an XML schema that has namespace urn:mpeg:mpegI:omaf:2020 and is specified as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:mpeg:mpegI:omaf:2020"
    xmlns:dash="urn:mpeg:dash:schema:mpd:2011"
    xmlns:omaf2="urn:mpeg:mpegI:omaf:2020"
    elementFormDefault="qualified">
<xs:import namespace="urn:mpeg:dash:schema:mpd:2011"
        schemaLocation="DASH-MPD.xsd"/>
<xs:import namespace=" urn:mpeg:mpegI:omaf:2017"
        schemaLocation="OMAFV1.xsd"/>
    <xs:simpleType name="listOfUnsignedShort">
     <xs:restriction>
         <xs:simpleType>
             <xs:list itemType="xs:unsignedShort"/>
         </xs:simpleType>
         <xs:minLength value="1"/>
     </xs:restriction>
    </xs:simpleType>
    <xs:element name="EntityGroup" type="omaf2:EntityGroupType"/>
    <xs:complexType name="EntityGroupType">
        <xs:sequence>
            <xs:element name="EntityIdList" type="omaf2:EntityIdType"
minOccurs="1"
maxOccurs="unbounded"/>
        </xs:sequence>
            <xs:attribute name=" group_type" type="dash:FourCCType"
                use="required"/>
            <xs:attribute name=" group_id" type="xs:unsignedInt"
use="required"/>
```

```
                <xs:attribute name="ref_overlay_id"
type="omaf2:listOfUnsignedShort"
                    use="optional"/>
            <xs:anyAttribute processContents="skip"/>
        </xs:complexType>
        <xs:complexType name="EntityIdType">
            <xs:attribute name="asid" type="xs:unsignedInt" use="required"/>
            <xs:attribute name="rsid" type="dash:StringNoWhitespaceType "
use="required"/>
            <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:complexType>
</xs:schema>
```

8.6 Segment formats

8.6.1 Initialization Segment for OMAF base track

8.6.1.1 General

The Initialization Segment for an OMAF base track shall conform to the Initialization Segment format for the ISO base media file format, as specified in ISO/IEC 23009-1. The Initialization Segment for an OMAF base track shall contain:

— A single stream header, containing a `FileTypeBox` and a `MovieBox` according to subclause 8.6.1.2.

— A single `TrackBox` for the OMAF base track according to subclause 8.6.1.3.

— A single `TrackBox` for each OMAF tile track referenced by the OMAF base track according to subclause 8.6.1.4.

8.6.1.2 Stream header

An Initialization Segment shall contain exactly one stream header, with ISOBMFF boxes and nesting, optionality and ordinality as specified in Table 30.

Table 30 — Stream header

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	Format Req.	Specification	Constraints	Description
`ftyp`									1	ISO/IEC 14496-12:2020, subclause 4.3		
Either `moov` or `!mov`									1	ISO/IEC 14496-12:2020, subclauses 8.2.1 and 8.19.6		
	`mvhd`								1	ISO/IEC 14496-12:2020, subclause 8.2.2	ISO/IEC 23000-19:2020, subclause 7.5.1	
	`mvex`								1	ISO/IEC 14496-12:2020, subclause 8.8.1		
		`mehd`							1	ISO/IEC 14496-12:2020, subclause 8.8.2	ISO/IEC 23000-19:2020, subclause 7.5.1	
		`trex`							1	ISO/IEC 14496-12:2020, subclause 8.8.3	ISO/IEC 23000-19:2020, subclause 7.5.14	

8.6.1.3 Track box format for OMAF base track

An Initialization Segment shall contain exactly one `TrackBox` for the OMAF base track, with ISOBMFF boxes and nesting, optionality and ordinality as specified in Table 31.

Table 31 — Track box for OMAF base track

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	Format Req.	Specification	Constraints	Description
	`trak`								1	ISO/IEC 14496-12:2020, subclause 8.3.1		
		`tkhd`							1	ISO/IEC 14496-12:2020, subclause 8.3.2	ISO/IEC 23000-19:2020, subclause 7.5.4 except that there are no constraints beyond those in ISO/IEC 14496-12 on the `width` and `height` fields and with the additional constraint that the field `matrix` shall always be set to the default values as defined in ISO/IEC 14496-12.	

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	Format Req.	Specification	Constraints	Description
		tref							1	ISO/IEC 14496-12:2020, subclause 8.3.3		Track references to OMAF tile tracks.
		mdia							1	ISO/IEC 14496-12:2020, subclause 8.4		
			mdhd						1	ISO/IEC 14496-12:2020, subclause 8.4.2	ISO/IEC 23000-19:2020, subclause 7.5.5	
			hldr						1	ISO/IEC 14496-12:2020, subclause 8.4.3		
			minf						1	ISO/IEC 14496-12:2020, subclause 8.4.4		
				vmhd					1	ISO/IEC 14496-12:2020, subclause 12.1.2	ISO/IEC 23000-19:2020, subclause 7.5.6	
				dinf					1	ISO/IEC 14496-12:2020, subclause 8.7.1		
					dref				1	ISO/IEC 14496-12:2020, subclause 8.7.2		
						snim			1	ISO/IEC 14496-12:2020, subclause 8.7.2.2		
				stbl					1	ISO/IEC 14496-12:2020, subclause 8.5	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stsd				1	ISO/IEC 14496-12:2020, subclause 8.5.2	ISO/IEC 23000-19:2020, subclause 7.5.10	
					stts				1	ISO/IEC 14496-12:2020, subclause 8.6.1.2	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stsc				1	ISO/IEC 14496-12:2020, subclause 8.7.4	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stco				1	ISO/IEC 14496-12:2020, subclause 8.7.5	ISO/IEC 23000-19:2020, subclause 7.5.12	

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	Format Req.	Specification	Constraints	Description
					stsz				1	ISO/IEC 14496-12:2020, subclause 8.7.3	ISO/IEC 23000-19:2020, subclause 7.5.12	

8.6.1.4 Track box format for OMAF tile track

An Initialization Segment shall contain exactly one `TrackBox` for each OMAF tile track referenced by the OMAF base track, with ISOBMFF boxes and nesting, optionality and ordinality as specified in Table 32.

Table 32 — Track box for OMAF tile track

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	Format Req.	Specification	Constraints	Description
	trak								+	ISO/IEC 14496-12:2020, subclause 8.3.1		
		tkhd							1	ISO/IEC 14496-12:2020, subclause 8.3.2	ISO/IEC 23000-19:2020, subclause 7.5.4 except that there are no constraints beyond those in ISO/IEC 14496-12 on the `width` and `height` fields and with the additional constraint that the field `matrix` shall always be set to the default values as defined in ISO/IEC 14496-12.	
		mdia							1	ISO/IEC 14496-12:2020, subclause 8.4		
			mdhd						1	ISO/IEC 14496-12:2020, subclause 8.4.2	ISO/IEC 23000-19:2020, subclause 7.5.5	
			hldr						1	ISO/IEC 14496-12:2020, subclause 8.4.3		
			minf						1	ISO/IEC 14496-12:2020, subclause 8.4.4		
				vmhd					1	ISO/IEC 14496-12:2020, subclause 12.1.2	ISO/IEC 23000-19:2020, subclause 7.5.6	
				dinf					1	ISO/IEC 14496-12:2020, subclause 8.7.1		

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	Format Req.	Specification	Constraints	Description
					dref				1	ISO/IEC 14496-12:2020, subclause 8.7.2		
						snim			1	ISO/IEC 14496-12:2020, subclause 8.7.2.2		
				stbl					1	ISO/IEC 14496-12:2020, subclause 8.5	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stsd				1	ISO/IEC 14496-12:2020, subclause 8.5.2	ISO/IEC 23000-19:2020, subclause 7.5.10	
					stts				1	ISO/IEC 14496-12:2020, subclause 8.6.1.2	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stsc				1	ISO/IEC 14496-12:2020, subclause 8.7.4	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stss				0/1	ISO/IEC 14496-12:2020, subclause 8.6.2	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stco				1	ISO/IEC 14496-12:2020, subclause 8.7.5	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stsz				1	ISO/IEC 14496-12:2020, subclause 8.7.3	ISO/IEC 23000-19:2020, subclause 7.5.12	

8.6.2 Tile Index Segment

This subclause specifies an Index Segment format for all the OMAF tile tracks associated with the same OMAF base track. Index Segments complying with this subclause are referred to as Tile Index Segments.

In the MPD, Tile Index Segments are indicated in the Representation carrying the OMAF base track. Each Tile Index Segment consists of a `SegmentTypeBox` and a `MovieFragmentBox` per each of the OMAF tile tracks. Tile Index Segments function as an index to Tile Data Segments and can be selectively retrieved by an OMAF player, depending on the current temporal and spatial position during playback.

The Tile Index Segment shall contain the ISOBMFF boxes with the nesting, optionality and ordinality as specified in Table 33.

Table 33 — Tile Index Segment

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	Format Req.	Specification	Constraints	Description
styp									1	ISO/IEC 14496-12:2020, subclause 8.16.2	Shall contain the brand 'sibm'.	
sidx									*	ISO/IEC 14496-12:2020, subclause 8.16.3	All 'sidx' boxes, when present, shall precede all 'moof' and '!mof' boxes.	
meof									*	Subclause 7.1.6		
Either moof or !mof									+	ISO/IEC 14496-12:2020, subclauses 8.8.4 and 8.19.7		
	mfhd								1	ISO/IEC 14496-12:2020, subclause 8.8.5		
	traf								+	ISO/IEC 14496-12:2020, subclause 8.8.6	sample_description_index shall reference a 'snim' box	
		tfhd							1	ISO/IEC 14496-12:2020, subclause 8.8.7	ISO/IEC 23000-19:2020, subclause 7.5.16 except that the base-data-offset-present flag may have any value and the default-base-is-moof flag shall be equal to 0.	
		tfdt							1	ISO/IEC 14496-12:2020, subclause 8.8.12		
		trun							1	ISO/IEC 14496-12:2020, subclause 8.8.8		

NOTE An OMAF player can resolve the composition timing information from the OMAF base track and does not need the composition timing from the OMAF tile tracks.

8.6.3 Tile Data Segment

This subclause specifies a Media Segment format for an OMAF tile track. Media Segments complying with this subclause are referred to as Tile Data Segments. Tile Data Segments may be used as the segment format of Representations carrying OMAF tile tracks.

The Tile Data Segment shall contain the ISOBMFF boxes with the nesting, optionality and ordinality as specified in Table 34.

Table 34 — Tile Data Segment

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	Format Req.	Specification	Constraints	Description
`styp`									1	ISO/IEC 14496-12:2020, subclause 8.16.2	Shall contain the brand `'imds'`.	
`imda`									+	ISO/IEC 14496-12:2020, subclause 8.1.4		

9 Omnidirectional media encapsulation and signalling in MMT

9.1 Architecture of MMT delivery in OMAF

Figure 29 depicts the reference architecture for OMAF content delivery over MMT:

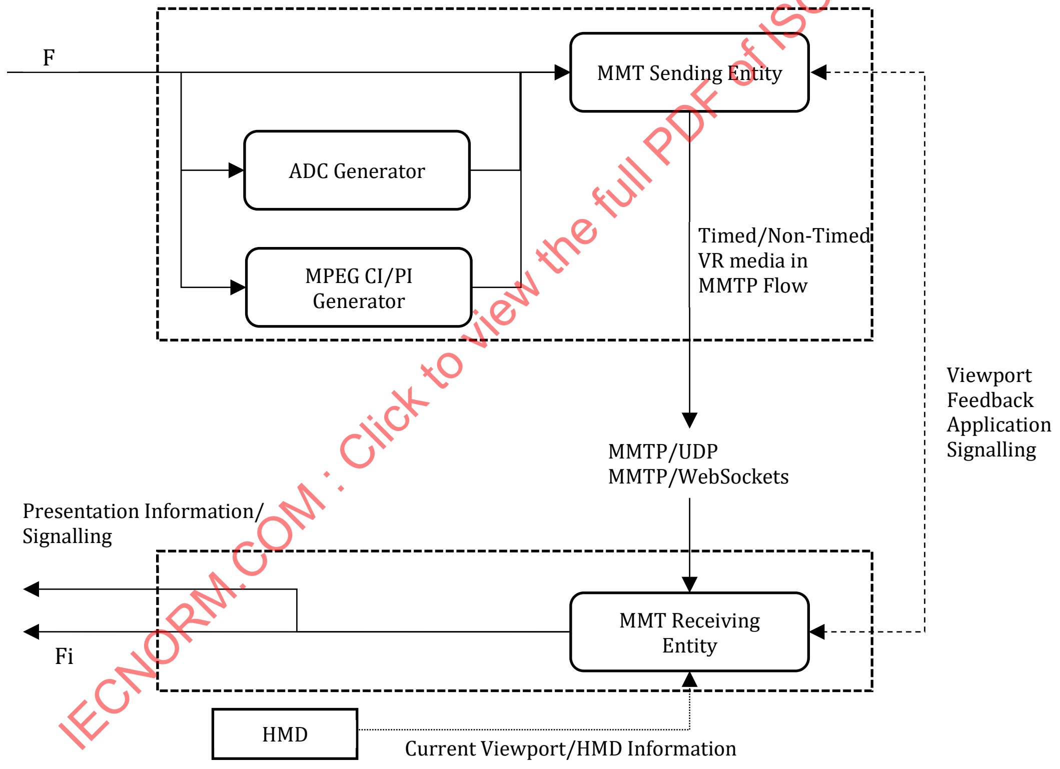

F
MMT Sending Entity
ADC Generator
MPEG CI/PI Generator
Timed/Non-Timed VR media in MMTP Flow
Viewport Feedback Application Signalling
MMTP/UDP
MMTP/WebSockets
Presentation Information/ Signalling
Fi
MMT Receiving Entity
HMD
Current Viewport/HMD Information

Figure 29 — Reference Architecture for OMAF over MMT

The OMAF content may be described in ADC (as defined in ISO/IEC 23008-1:2017, subclause 6.5) to assist the MMT sending entity during the streaming process. The Presentation Information should contain information to describe MPUs that are conformant to OMAF to enable appropriate processing by the application.

MMT delivery may use the MMTP protocol over UDP, MMTP/WebSockets/TCP, or another alternative.

The player receives information about the current viewing direction and viewport and the HMD characteristics. Only a portion of the 360 video is needed at a time depending on the current viewing direction. Field-of-view streaming (FoV) or alternatively called as view-dependent streaming is used to reduce the bandwidth needs during streaming. When MMT is used, view-dependent streaming may be achieved using one of the following two approaches:

— Client-based approach: the MMT receiving entity may be instructed by the player to select a subset of the Assets that carry video data covering the current viewport. MMT session control procedures as defined in ISO/IEC 23008-1 shall be used to request the selected set of Assets from the MMT sending entity. The player uses the VR application specific signalling message to convey the content coverage and the region-wise packing information in accordance with the definitions in subclauses 7.5.3, 7.6 and 9.3, to select appropriate Asset to switch to for view-dependent streaming.

— Server-based approach: the MMT receiving entity relies on the MMT sending entity to select the correct subset of Assets that provide video content to cover the current viewport. The receiving entity uses the VR application-specific signalling as described in subclause 9.3 to send information about the current viewport to the sending entity.

9.2 OMAF signalling in MPEG composition information

MMT supports MPEG CI as the Presentation Information that describes the media presentation. In MPEG CI, OMAF Assets shall be mapped to a to an HTML5 Canvas element instead of an HTML5 video element as shown in the following example:

```
<MediaSync begin="T10M5S" dur="T32M" refId="canvas1">
      <sourceList componentId="VRvideo">
            <mediaSrc
mimeType='video/mp4;codecs="encv.cbcs.resv.podv+erpv.hvc1"'>
media1.mp4</mediaSrc>
      </sourceList>
</MediaSync>
```

The `codecs` MIME type parameter shall be set according to ISO/IEC 14496-12.

The user agent should decode the OMAF Asset in the background, e.g. using a hidden HTML5 video element and render the omnidirectional media to the referenced HTML5 Canvas element.

For view-dependent media streaming and playback, the media data should be retrieved using a protocol that supports view-dependent streaming of VR content, such as using MMT with VR specific signalling (see subclause 9.3). In such case, an MMT session entry point should be used as the mediaSrc element.

9.3 VR application-specific MMT signalling

9.3.1 General

MMT, specified in ISO/IEC 23008-1, defines an application-specific signalling message that allows for the delivery of application-specific information. For the purpose of streaming VR content that is formatted according to the OMAF specification, a VR specific Asset descriptor, a stereo video Asset descriptor, and a VR application-specific signalling message are defined. The VR application-specific signalling message shall have an application identifier with a URN of value of "`urn:mpeg:mmt:app:vr:2017`" that is specified in ISO/IEC 23008-1.

A new Asset descriptor for OMAF VR formatted content is defined under the name VR Information Asset descriptor. The VR Information descriptor shall be present in all Assets that carry OMAF formatted content. A stereo video Asset descriptor is also defined, and shall be present in all Assets that carry OMAF VR formatted stereoscopic content.

In the specified VR signalling message, the following set of application message types are defined:

— VRViewDependentSupportQuery: the client uses this command to discover if the server supports view-dependent streaming

— VRViewDependentSupportResponse: the server replies with an indication of its support capability for view-dependent streaming.

— VRViewportChangeFeedback: the receiving entity sends an indication of the current viewport to the sending entity.

— VRViewDependentAssetsInformation: upon determining the set of OMAF Assets that match the requested viewport, the sending entity sends this message to inform the client about the new OMAF Assets that will be streamed to the receiving entity.

The VRViewportChangeFeedback and VRViewDependentAssetsInformation message are used together to support server-driven view-dependent streaming of OMAF Assets. The content selection message is used to support the client-driven view-dependent streaming scenario, respectively.

To support guided rendering, where the renderer follows an indicated Region of Interest, or to follow the recommended viewport timed metadata track of OMAF, the VRROIGuide application message type is defined.

The list of defined application message types is provided in Table 35.

Table 35 — VR application message types

Application Message Type	Application Message Name
0x01	VRViewDependentSupportQuery
0x02	VRViewDependentSupportResponse
0x03	VRViewportChangeFeedback
0x04	VRViewDependentAssetInformation
0x05	VRROIGuide
0x06	VR3DAudioAssetInformation
0x07-0xFF	Reserved for future use

9.3.2 MMT signalling

9.3.2.1 VRInformation asset descriptor

9.3.2.1.1 General

This Asset descriptor is used to inform the receiving entity and the VR application about the content of the current Asset that carries VR content. The information describes the projection type that is used, how the VR content is region-wise frame packed, and what areas on the sphere it covers. The indication if content is stereoscopic with frame packing is provided through a separate Asset descriptor.

9.3.2.1.2 Syntax

The syntax of the VR Information Asset descriptor is shown in Table 36.

Table 36 — VR Information Asset descriptor syntax

Syntax	Value	No. of bits	Mnemonic
VR_information_descriptor() {			
descriptor_tag		16	uimsbf
descriptor_length		16	uimsbf
rwfp_flag		1	bslbf
srqr_flag		1	bslbf
2dqr_flag		1	bslbf
viewingspace_info_flag		1	bslbf
viewpoint_flag		1	bslbf
reserved	'111'	3	bslbf
ProjectionFormatStruct()			
InitialViewingOrientationSample()			
ContentCoverageStruct()			
if(rwfp_flag == 1) {			
RegionWisePackingStruct()			
}			
if(srqr_flag == 1) {			
SphereRegionQualityRankingStruct()			
}			
if(2dqr_flag == 1) {			
2DRegionQualityRankingStruct()			
}			
if(viewingspace_info_flag == 1) {			
ViewingSpaceStruct()			
}			
if(viewpoint_flag == 0) {			
viewpoint_id		32	uimsbf
ViewpointInformationStruct(0,0)			
viewpoint_label_length		8	uimsbf
for(i=0; i < viewpoint_label_length; i++)			
viewpoint_label_byte[i]		8	uimsbf
viewpoint_groupdescr_length		8	uimsbf
for(i=0; i < viewpoint_groupdescr_length; i++)			
viewpoint_group_description_byte[i]		8	uimsbf
if(switching_info_present_flag) {			
for(j=0; j < num_viewpoint_switching; j++){			
if(transition_effect_flag && transition_effect_type == 6) {			
transitioneffect_scheme_uri_length[j]		8	uimsbf
for(i=0;i<transitioneffect_scheme_uri_length[j]; i++) transition_effect_scheme_uri_byte[j][i]		8	uimsbf
transitioneffect_scheme_value_length[j]		8	uimsbf

Syntax	Value	No. of bits	Mnemonic
for(i=0;i<transitioneffect_scheme_value_length[j]; i++)			
transition_effect_scheme_value_byte[j][i]		8	uimsbf
}			
}			
}			
}			

9.3.2.1.3 Semantics

descriptor_tag indicates the type of a descriptor.

descriptor_length specifies the length in bytes counting from the next byte after this field to the last byte of the descriptor.

rwfp_flag equal to 1 indicates that region-wise frame packing has been applied to the content of this Asset and that the RegionWisePackingStruct() that describes it is present.

srqr_flag equal to 1 indicates that sphere region quality information is present.

2dqr_flag equal to 1 indicates that 2D region quality information is present.

viewingspace_info_flag equal to 1 indicates that viewing space information as specified by ViewingSpaceStruct() is present in this VR_information_descriptor(). Viewingspace_info_flag equal to 0 indicates that viewing space information as specified by ViewingSpaceStruct() is not present in this VR_information_descriptor().

viewpoint_flag equal to 0 indicates that content in the corresponding asset contains viewpoint information corresponding to the viewpoint identifiable by viewpoint_id. When viewpoint_flag is equal to 1, the asset contains only one viewpoint.

ProjectionFormatStruct() provides information on the projection format that is used. ProjectionFormatStruct() is identical to the definition in subclause 7.5.2.

InitialViewingOrientationSample() provides information about the current initial viewing orientation. InitialViewingOrientationSample() is identical to the definition in subclause 7.7.4.

ContentCoverageStruct() indicates the sphere region(s) covered by the track. ContentCoverageStruct() is identical to the definition in subclause 7.5.5.

RegionWisePackingStruct() indicates that the projected pictures are packed region-wise and require unpacking prior to rendering, according to the region-wise packing process information as indicated. RegionWisePackingStruct() is identical to the definition in subclause 7.5.3.

SphereRegionQualityRankingStruct() indicates a relative quality order of quality ranking sphere regions. SphereRegionQualityRankingStruct() is identical to the definition in subclause 7.8.2.

`2DRegionQualityRankingStruct()` indicates a relative quality order of quality ranking 2D regions. `2DRegionQualityRankingStruct()` is identical to the definition in subclause 7.8.3.

`ViewingSpaceStruct()` indicates viewing space information. `ViewingSpaceStruct()` is identical to the definition in subclause 7.14.1.

`viewpoint_id` specifies the identifier of a viewpoint.

`ViewpointInformationStruct(0, 0)` provides information about the viewpoint represented by this asset as specified in subclause 7.12.2.2.

`viewpoint_label_length` specifies the length in bytes of the human readable text label for the viewpoint with identifier specified by `viewpoint_id`.

`viewpoint_label_byte[i]` specifies the i-th UTF-8 character of the viewpoint label string.

`viewpoint_groupdescr_length` specifies the length in bytes of the human readable text label for the description of a viewpoint group for the viewpoint specified by `viewpoint_id`.

`viewpoint_group_description_byte[i]` specifies the i-th UTF-8 character of the description of a viewpoint group.

`transitioneffect_scheme_uri_length[j]` specifies the length of j-th transition effect scheme URI element in bytes. `transitioneffect_scheme_uri_length[j]` shall not be equal to 0 when transition_effect_type is equal to 6.

`transition_effect_uri_byte[j][i]` specifies the i-th UTF-8 character of the j-th transition effect scheme URI. For each j, overall transition effect scheme URI assembled from concatenating `transition_effect_scheme_uri_byte[j][i]` for each i, shall be formatted as an absolute URI as specified in IETF Internet Standard 66.

`transitioneffect_scheme_value_length[j]` specifies the length of j-th transition effect scheme value element in bytes.

`transition_effect_scheme_value_byte[j][i]` specifies the i-th UTF-8 character of the j-th transition effect scheme value.

9.3.2.2 VRViewDependentSupportQuery

9.3.2.2.1 Syntax

The syntax of the VRViewDependentSupportQuery is shown in Table 37.

Table 37 — VRViewDependentSupportQuery

Syntax	Value	No. of bits	Mnemonic
`Application() {`			
message_id		16	uimsbf
version		8	uimsbf
length		16	uimsbf
`message_payload{`			
application_identifier()			
`if(application_identifier == "urn:mpeg:mmt:app:vr:2017")`			
`{`			
app_message_type		8	uimsbf
`if(app_message_type == 0x01) {`			
hmd_hor_resolution		16	uimsbf
hmd_ver_resolution		16	uimsbf
hmd_hor_fov		32	uimsbf
hmd_ver_fov		32	uimsbf
`}`			
`}`			
`}`			
`}`			

9.3.2.2.2 Semantics

`message_id` indicates the identifier of the VRViewDependentSupportQuery message.

`version` indicates the version of VRViewDependentSupportQuery message.

`length` indicates the length of VRViewDependentSupportQuery message in bytes, counting from the beginning of the next field to the last byte of the VRViewDependentSupportQuery message. The value of this field shall not be equal to 0.

`application_identifier` indicates the application identifier as a urn that uniquely identifies the application to consume the contents of this message.

`app_message_type` defines an application-specific message type provided in Table 35.

`hmd_hor_resolution` and `hmd_ver_resolution` provides the horizontal and vertical resolution of the display of the HMD in units of square pixels.

`hmd_hor_fov` and `hmd_ver_fov` provide the horizontal and vertical field of view of the HMD, in units of 2^{-16} degrees. `hmd_hor_fov` shall be in the range of 0 to 360 * 2^{16}, inclusive. `hmd_ver_fov` shall be in the range of 0 to 180 * 2^{16}, inclusive.

9.3.2.3 VRViewDependentSupportResponse

9.3.2.3.1 Syntax

The syntax of the ViewDependentSupportResponse is shown in Table 38.

Table 38 — VRViewDependentSupportResponse syntax

Syntax	Value	No. of bits	Mnemonic
`Application() {`			
message_id		16	uimsbf
version		8	uimsbf
length		16	uimsbf
`message_payload{`			
application_identifier()			
`if(application_identifier == "urn:mpeg:mmt:app:vr:2017")`			
`{`			
app_message_type		8	uimsbf
`if(app_message_type == 0x02) {`			
view_dependent_support		1	bslbf
reserved	'1111111'	7	uimsbf
`}`			
`}`			
`}`			
`}`			

9.3.2.3.2 Semantics

`message_id` indicates the identifier of the VRViewDependentSupportResponse message.

`version` indicates the version of VRViewDependentSupportResponse message.

`length` indicates the length of VRViewDependentSupportResponse message in bytes, counting from the beginning of the next field to the last byte of the VRViewDependentSupportResponse message. The value of this field shall not be equal to 0.

`application_identifier` indicates the application identifier as a urn that uniquely identifies the application to consume the contents of this message.

`app_message_type` defines an application-specific message type provided in Table 35.

`view_dependent_support` equal to 1 indicates that view-dependent streaming is supported by the server. `view_dependent_support` equal to 0 indicates that view dependent streaming is not supported by the server.

9.3.2.4 VRViewportChangeFeedback

9.3.2.4.1 General

The MMT VR receiving entity feedbacks the virtual camera direction information periodically or in case of FOV changing event to the MMT sending entity to inform about the current VR virtual camera direction.

9.3.2.4.2 Syntax

The syntax of the VRViewportChangeFeedback is shown in Table 39.

Table 39 — VRViewportChangeFeedback

Syntax	Value	No. of bits	Mnemonic
`Application() {`			
message_id		16	uimsbf
version		8	uimsbf
length		16	uimsbf
`message_payload{`			
application_identifier()			
`if(application_identifier == "urn:mpeg:mmt:app:vr:2017")`			
`{`			
app_message_type		8	uimsbf
`if(app_message_type == 0x03) {`			
dirx		16	uimsbf
diry		16	uimsbf
dirz		16	uimsbf
last_processed_media_timestamp		64	uimsbf
`}`			
`}`			
`}`			
`}`			

9.3.2.4.3 Semantics

`message_id` indicates the identifier of the VRViewportChangeFeedback message.

`version` indicates the version of VRViewportChangeFeedback message.

`length` indicates the length of VRViewportChangeFeedback message in bytes, counting from the beginning of the next field to the last byte of the VRViewportChangeFeedback message. The value of this field shall not be equal to 0.

`application_identifier` indicates the application identifier as a urn that uniquely identifies the application to consume the contents of this message.

`app_message_type` defines an application-specific message type provided in Table 35.

`dirx`, `diry`, and `dirz` define the x, y, and z component, respectively, of the three-dimensional viewing direction unit vector in a Cartesian coordinate system with (x, y, z) equal to (1, 0, 0) corresponding to the sphere location with (ϕ, θ) equal to (0, 0). The value of `dirx`, `diry`, or `dirz` shall be in the range of 1 to 65535, inclusive, where 1 corresponds to −1, 32768 corresponds to 0, and 65535 corresponds to +1.

The fields `dirx`, `diry`, and `dirz` may be calculated according to the azimuth ϕ and elevation θ, which may be obtained from HMD sensor. Herein, the viewing direction denotes a three-dimensional vector from the centre of the sphere pointing to a location on the surface of the sphere. An example of specifying `dirx`, `diry`, and `dirz` is given in Clause F.2.

`last_processed_media_timestamp` indicates the presentation timestamp of the last media unit that has been appended to the decoder buffer. This field is used by the MMT sending entity to determine the next media unit from the new asset that is sent to the OMAF player. The next media unit is the one with a timestamp or sequence number immedialy following the indicated timestamp. The MMT sending entity switches from transmitting the old asset (representing the old viewport) to transmitting the new asset (representing the new viewport) starting from the following media timestamp, in order to reduce the delay of receiving the new viewport. This means that the transmitted media data from the new asset may overlap in media time with already transmitted media data from the old asset. If more than one OMAF Asset is being delivered, the minimum value shall be used.

9.3.2.5 VRViewDependentAssetInformation

9.3.2.5.1 General

This signalling is used to indicate to the receiving entity that MMT packets belonging to the new asset as specified by the packet_id will be sent to the receiving entity from the sending entity. Upon receiving this signal, the next fragment received by the client belongs to the new Asset as indicated by the packet_id. Based on the Asset ID, the renderer is able to correctly associate the received media with the coverage information and to correctly render the omnidirectional video using the video texture from the new asset.

9.3.2.5.2 Syntax

The syntax of the VRViewDependentAssetInformation is shown in Table 40.

Table 40 — VRViewDependentAssetInformation syntax

Syntax	Value	No. of bits	Mnemonic
`Application() {`			
message_id		16	uimsbf
version		8	uimsbf
length		16	uimsbf
`message_payload{`			
application_identifier()			
`if(application_identifier == "urn:mpeg:mmt:app:vr:2017")`			
`{`			
app_message_type		8	uimsbf
`if(app_message_type == 0x04) {`			
reserverd	'1111 1111'	8	bslbf
packet_id		16	uimsbf
ContentCoverageStruct()			
`}`			
`}`			
`}`			
`}`			

9.3.2.5.3 Semantics

`message_id` indicates the identifier of the VRViewDependentAssetInformation message.

`version` indicates the version of VRViewDependentAssetInformation message.

`length` indicates the length of VRViewDependentAssetInformation message in bytes, counting from the beginning of the next field to the last byte of the VRViewDependentAssetInformation message. The value of this field shall not be equal to 0.

`application_identifier` indicates the application identifier as a urn that uniquely identifies the application to consume the contents of this message.

`app_message_type` defines an application-specific message type provided in Table 35.

`packet_id` indicates the packet_id that is associated with the OMAF Asset. This field is passed to the renderer to indicate the Asset ID of the asset that is currently being received and played in high resolution. Based on the Asset ID, the renderer is able to determine the coverage information of the video texture that corresponds to the current Asset.

`ContentCoverageStruct()` indicates the sphere region(s) covered by the content. `ContentCoverageStruct()` is identical to the definition in subclause 7.5.5.

9.3.2.6 VRROIGuide

9.3.2.6.1 General

MMT sending entity sends VRROIGuide message to guide what to display from the delivered entire video content. If the MMT receiving entity receives the media data of the entire video, only the region specified by the VRROIGuide message shall be presented.

9.3.2.6.2 Syntax

The syntax of VRROIGuide message is defined in Table 41.

Table 41 — VRROIGuide syntax

Syntax	Value	No. of bits	Mnemonic
`Application() {`			
`    message_id`		16	uimsbf
`    version`		8	uimsbf
`    length`		16	uimsbf
`    message_payload{`			
`        application_identifier()`			
`        if(application_identifier == "urn:mpeg:mmt:app:vr:2017")`			
`        {`			
`            app_message_type`		8	uimsbf
`            if(app_message_type == 0x05) {`			
`                guide_type`		1	uimsbf
`                reserved`	'1111 1111'	7	bslbf
`                region_type`		8	uimsbf
`                guide_region(){`			
`                  if ((region_type == 0x01) || (region_type == 0x03)) {`			

guide_shape_type		8	uimsbf
SphereRegionStruct(1, 1)			
}			
else if ((region_type == 0x02) || (region_type == 0x04)) {			
centre_x		16	uimsbf
centre_y		16	uimsbf
width		16	uimsbf
height		16	uimsbf
}			
}			
guide_start		32	uimsbf
guide_duration		32	uimsbf
}			
}			
}			

9.3.2.6.3 Semantics

`message_id` indicates the identifier of the VRROIGuide message.

`version` indicates the version of VRROIGuide message.

`length` indicates the length of VRROIGuide message in bytes, counting from the beginning of the next field to the last byte of the VRROIGuide message. The value of this field shall not be equal to 0.

`application_identifier` indicates the application identifier as a urn that uniquely identifies the application to consume the contents of this message.

`app_message_type` defines an application-specific message type provided in Table 35.

`guide_type` indicates whether presenting only the guide region specified by `guide_region` is mandatory. If this flag is set to 1, only the guide region should be displayed. If this flag is set to 0, both the guide region and the rest of region may be displayed.

`region_type` defines the type of guide region. The values for this field are specified in Table 42.

Table 42 — Value of `region_type`

Type	Description
0x00	reserved
0x01	centre point of a viewport on sphere
0x02	centre point of a viewport on frame
0x03	viewport on sphere
0x04	viewport on frame
0x05~0xFF	reserved

`guide_region` specifies the guide region to be displayed. If the value of `region_type` is 0x01 or 0x03, the guide region is specified as the defintion of `SphereRegionStruct()` in subclause 7.5.6.

For the `SphereRegionStruct(1, 1)` when included in the `guide_region()`, `interpolate` shall be equal to 0.

`guide_shape_type` specifies the shape of the guide region. `guide_shape_type` has the same semantics as `shape_type` specified in clasue 7.7.2.3.

`centre_x` specifies the horizontal centre positon of the guide region in pixels.

`centre_y` specifies the vertical centre positon of the guide region in pixels.

`width` specifies the width of the guide region in pixels.

`height` specifies the height of the guide region in pixels.

`guide_start` indicates the presentation time of media data where the region guide starts. This field is a UTC time in NTP format and 32 bits long.

`guide_duration` indicates the duration the region guide from the time indicated by `guide_start`. This field is expressed in milliseconds.

9.3.2.7 Stereo video asset descriptor

9.3.2.7.1 Syntax

The syntax of the Stereo Video Asset descriptor is shown in Table 43.

Table 43 — Stereo video asset descriptor

Syntax	Value	No. of bits	Mnemonic
Stereo_video_descriptor(){			
descriptor_tag		16	uimsbf
descriptor_length		8	uimsbf
StereoVideoBox()			
}			

9.3.2.7.2 Semantics

`descriptor_tag` indicates the type of a descriptor.

`descriptor_length` specifies the length in bytes counting from the next byte after this field to the last byte of the descriptor.

`StereoVideoBox()` provides a copy of the `StereoVideoBox` as defined in ISO/IEC 14496-12.

9.3.2.8 VR3DAudioAssetInformation

9.3.2.8.1 General

The VR3DAudioAssetInformation message shall be conveyed for all Assets that carry OMAF formatted content compliant to the OMAF 3D audio baseline profile.

9.3.2.8.2 Syntax

The syntax of **VR3DAudioAssetInformation** message is defined in Table 44.

Table 44 — VR3DAudioAssetInformation syntax

Syntax	Value	No. of bits	Mnemonic
VR3DAudioAssetInformation() {			
message_id		16	uimsbf
version		8	uimsbf
length		16	uimsbf
message_payload {			
application_identifier()			
if(application_identifier=="urn:mpeg:mmt:app:vr:2017") {			
app_message_type		8	uimsbf
if(app_message_type == 0x06) {			
number_of_assets		8	uimsbf
for (i=0; i<number_of_assets; i++) {			
asset_id_length		32	uimsbf
for (j=0; j<asset_id_length; j++) {			
asset_id_byte		8	uimsbf
}			
codec_code		4*8	uimsbf
profile_level_indication		8	uimsbf
num_preselections		8	uimsbf
channel_configuration		6	uimbsf
multi_stream_info_present		1	bslbf
reserved	'1'	1	bslbf
for (j=0; j<num_preselections; j++) {			
preselection_id		8	uimbsf
interactivity_enabled		1	bslbf
language_present		1	bslbf
accessibility_role_present		1	bslbf
label_present		1	bslbf
reserved	'1111'	4	bslbf
if (language_present) {			
num_languages_minus1		8	uimbsf
for(k=0;k<num_languages_minus1+1;k++) {			
language_length		8	uimbsf
for (l=0;l< language_length;l++) {			
language_byte		8	uimbsf
}			
}			
}			
if (accessibility_role_present) {			
for(k=0;k<num_languages_minus1+1;k++) {			
accessibility		8	uimbsf
}			
role		8	uimbsf
}			
if (label_present) {			
label_length		8	uimbsf
for (k=0; k< label_length; k++) {			
label_data_byte		8	uimbsf

Syntax	Value	No. of bits	Mnemonic
`                }`			
`            }`			
`            if (multi_stream_info_present) {`			
`                **preselection_aux_stream_info()**`			Table 46
`            }`			
`        } /* end of for num_preselections loop */`			
`        if (multi_stream_info_present) {`			
`            **multi_stream_info()**`			Table 47
`        }`			
`    } /* end of for number_of_assets loop*/`			
`  } /* end of if(app_message_type == 0x06)*/`			
`  }`			
` }`			
`}`			

9.3.2.8.3 Semantics

`message_id` indicates the identifier of the VR3DAudioAssetInformation message.

`version` indicates the version of VR3DAudioAssetInformation message.

`length` indicates the length of VR3DAudioAssetInformation message in bytes, counting from the beginning of the next field to the last byte of the VR3DAudioAssetInformation message. The value of this field shall not be equal to 0.

`application_identifier` indicates the application identifier as an urn that uniquely identifies the application to consume the contents of this message.

`app_message_type` defines an application-specific message type provided in Table 35.

`number_of_assets` specifies the number of audio assets described by this descriptor.

`asset_id_length` specifies the length in bytes of the audio asset ID.

`asset_id_byte` contains a byte of the audio asset ID.

`codec_code` specifies the 4-character code for MPEG-H 3D Audio. The value of these four characters shall be one of `'mhm1'` or `'mhm2'` with a semantic meaning for these codes as specified in ISO/IEC 23008-3.

`profile_level_indication` indicates the audio profile and level of the associated preselection and shall contain an `mpegh3daProfileLevelIndication` field as specified in ISO/IEC 23008-3:2019, subclause 5.3.2.

`num_preselections` indicates the number of Preselections that are available within the main stream and all auxiliary streams. The minimum number of `num_preselections` shall be '1' for the main stream. For auxiliary streams `num_preselections` shall have the value '0' so that for auxiliary streams no preselection information is present in the descriptor. This field contains a `mae_numGroupPresets` field as specified in ISO/IEC 23008-3:2019, subclause 15.3.

`channel_configuration` specifies the Channel Configuration and shall have the same value as the `ChannelConfiguration` field specified in ISO/IEC 23091-3. Valid values are 1-7,9-12, 14-17 or 19.

`multi_stream_info_present` equal to 1 indicates that the elements in the `multi_stream_info()` structure are present.

`preselection_id` identifies the ID of this Preselection. The first preselection in the loop shall have the lowest preselection_id and shall be the default Preselection. This field indicates the `mae_GroupPresetID` field as specified in ISO/IEC 23008-3:2019, subclause 15.3.

`interactivity_enabled` equal to 1 indicates that that the audio preselection contains elements with associated metadata, which enable user interactivity.

`language_present` equal to 1 indicates that language information for this Preselection is present.

`accessibility_role_present` equal to 1 indicates that accessibility and role information for this Preselection is present.

`label_present` equal to 1 indicates that a text label for this Preselection is present.

`num_languages_minus1` plus 1 specifies the number of languages that are available within this Preselection. When not present the value of num_languages_minus1 shall be inferred to be equal to 0.

`language_length` specifies the length in bytes of each language supported in the Preselection. The first language in the loop (k is equal to 0) shall be the primary language for the Preselection. The remaining language(s) in the loop (k is not equal to 0) shall indicate the additional language(s) available in the Preselection.

`language_byte` contains a UTF-8 character of the k-th language of the Preselection. The language of the Preselection shall be given by a language tag as defined by IETF BCP 47, *Tags for Identifying Languages*. The language indicated by this field should correspond to the information conveyed in `mae_contentLanguage` of the default dialog element: the `maeGroup` which is marked as default in `mae_switchGroupDefaultGroupID` and is tagged in `mae_contentKind` as dialog. This information is carried in the `AudioSceneInformation()` of the MPEG-H Audio stream as specified in ISO/IEC 23008-3.

`accessibility` identifies the accessibility support for each language in this Preselection. Table 45 specifies the bit used to indicate if the Preselection contains support for a particular audio accessibility service. When one bit specified in Table 45 is set to '1' it indicates the Preselection contains the corresponding audio accessibility service.

Table 45 — Accessibility bits

Bit	Audio Accessibility Service
0 (MSB)	For Visually Impaired (Video Description Service)
1	Dialog Enhancement enabled
2	Emergency information
3-7	Reserved zero bits
Reserved bits shall be set to zero.	

The setting of the bits in the accessibility field should correspond to the `mae_groupPresetKind` value in the `mae_GroupPresetDefinition()` structure and the `mae_contentKind` values in the `mae_ContentData()` structures in the `AudioSceneInformation()` of the MPEG-H 3D Audio stream as specified in ISO/IEC 23008-3. The mapping from the MPEG-H Audio metadata fields should be done as follows:

— Bit 0 should be set to '1', if the `mae_contentKind` value of at least one Audio Element is set to '9' ("audio description/visually impaired").

— Bit 1 should be set to '1', if at least the dialog Audio Elements with a `mae_contentKind` value of '2' ("dialogue") have `mae_allowGainInteractivity` set to '1' and `mae_interactivityMaxGain` set to a non-zero value in the corresponding `mae_GroupDefinition()` structure.

— Bit 2 should be set to '1', if the `mae_contentKind` value of at least one Audio Element is set to '12' ("emergency").

`role` indicates the role or service type of the Preselection. The role values shall correspond to the role scheme `@schemeIdUri` equal to `"urn:mpeg:dash:role:2011"` defined in ISO/IEC 23009-1. For a description of the `role` values, see ISO/IEC 23009-1:2019, subclause 5.8.5.5.

`label_length` specifies the length in bytes of this Preselection text label.

`label_data_byte` contains a UTF-8 character of the Preselection text label.

Table 46 — Syntax for preselection_aux_stream_info()

Syntax	Value	No. of bits	Mnemonic
preselection_aux_stream_info() {			
num_preselection_aux_streams		8	uimbsf
for (m=0; m<num_preselection_aux_streams; m++) {			
aux_stream_id		8	uimbsf
}			
}			

`num_preselection_aux_streams` indicates the number of auxiliary streams that are required for this specific Preselection.

`aux_stream_id` ientifies the ID of the auxiliary stream that is required for this specific Preselection.

Table 47 — Syntax for multi-stream_info()

Syntax	Value	No. of bits	Mnemonic
multi_stream_info() {			
this_is_main_stream		1	bslbf
this_stream_id		7	uimbsf
reserved	'1'	1	bslbf
bundle_id		7	uimbsf
if (this_is_main_stream) {			
reserved	'1'	1	bslbf
num_auxiliary_streams		7	uimbsf
for (m=0; m<num_auxiliary_streams; m++) {			
reserved	'1'	1	bslbf
auxiliary_stream_id		7	uimbsf
}			
}			
}			

`this_is_main_stream` equal 1 indicates that this stream contains a main stream that may be presented on its own, or that may be combined with additional audio components from an auxiliary stream. The main stream shall be delivered as an MMTP/MPU stream. Auxiliary streams are delivered as MMTP/MPU streams using different asset_IDs and they are signaled within the VR3DAudioAssetInformation message.

`this_stream_id` indicates the ID of this audio stream. This ID shall be unique within one bundle, i.e. for all streams that have the same `bundle_id`.

`bundle_id` identifies a unique ID for one bundle of audio streams. A bundle consists of exactly one main stream and one or more additional auxiliary streams that shall have the same `bundle_id`. The auxiliary streams contain additional audio components that may be combined with the main stream.

`num_auxiliary_streams` indicates the number of auxiliary streams that are available to be combined with the main stream.

`auxiliary_stream_id` identifies the ID of the auxiliary stream. The ID of all auxiliary streams shall be unique within one bundle.

9.3.2.9 VR fisheye information asset descriptor

9.3.2.9.1 General

This asset descriptor is used to inform the receiving entity and the VR application about the content of the current asset that carries VR fisheye content. The information describes the essential fisheye video parameters for enabling stitching and rendering at the receiving entity.

9.3.2.9.2 Syntax

The syntax of the VR fisheye information asset descriptor is shown in Table 48.

Table 48 — VR fisheye information asset descriptor syntax

Syntax	Value	No. of bits	Mnemonic
`VR_fisheye_information_descriptor() {`			
descriptor_tag		16	uimsbf
descriptor_length		16	uimsbf
FisheyeVideoEssentialInfoStruct()			
`}`			

9.3.2.9.3 Semantics

`descriptor_tag` indicates the type of a descriptor.

`descriptor_length` specifies the length in bytes counting from the next byte after this field to the last byte of the descriptor.

`FisheyeVideoEssentialInfoStruct()` provides information about the fisheye video content of the current asset. `FisheyeVideoEssentialInfoStruct()` is identical to the definition in subclause 6.2.

10 Media profiles

10.1 Video profiles

10.1.1 Overview

Subclause 10.1 defines media profiles for video. Table 49 provides an overview of the supported features. The detailed, specification for each video profile is subsequently provided in the referenced subclause.

Table 49 — Overview of OMAF media profiles for video

Media Profile	Codec	Profile	Level	Scheme Types	Brand	Subclause
HEVC-based viewport-independent OMAF video profile	HEVC	Main 10	5.1	`podv` and `erpv`	`hevi`	10.1.2
Unconstrained HEVC-based viewport-independent OMAF video profile	HEVC	Main 10	any	`podv` and `erpv`	`uhvi`	10.1.5
HEVC-based viewport-dependent OMAF video profile	HEVC	Main 10	5.1	`podv` and at least one of `erpv` and `ercm`	`hevd`	10.1.3
AVC-based viewport-dependent OMAF video profile	AVC	Progressive High	5.1	`podv` and at least one of `erpv` and `ercm`	`avde`	10.1.4
Advanced tiling OMAF video profile	HEVC	Main 10	any	`modv` and `meov`	`adti`	10.1.6
Simple tiling OMAF video profile	HEVC	Main 10	any	`podv` and at least one of `erpv` and `ercm`	`siti`	10.1.7

NOTE For the HEVC Main 10 profile, the bit depth of decoded pictures can be either 8 bits or 10 bits.

10.1.2 HEVC-based viewport-independent OMAF video profile

10.1.2.1 General

This OMAF video profile can be described as follows:

— Both monoscopic and stereoscopic spherical video up to 360 degrees are supported.

— The profile requires neither viewport-dependent delivery nor viewport-dependent decoding.

— Regular HEVC encoders, DASH packagers, DASH clients, file format parsers, and HEVC decoder engines that do not need special features for handling of viewport-dependent delivery and decoding can be used for encoding, distribution and decoding.

10.1.2.2 Elementary stream constraints

The elementary stream constraints apply to the HEVC bitstream that is reconstructed from a file as specified in subclause 10.1.2.5.

The bitstream shall comply with HEVC Main 10 profile, Main tier, Level 5.1.

All pictures shall be encoded as coded frames, and shall not be encoded as coded fields.

All the active SPSs of the bitstream shall be constrained as follows:

— general_progressive_source_flag shall be equal to 1.

— general_frame_only_constraint_flag shall be equal to 1.

— general_interlaced_source_flag shall be equal to 0.

When VUI is present, aspect_ratio_idc should not be present or aspect_ratio_idc should be equal to 0 (unspecified) or 1 (square).

NOTE When aspect_ratio_idc is not present, Rec. ITU-T H.265 | ISO/IEC 23008-2 specifies that aspect_ratio_idc is inferred to be equal to 0.

For each picture, there shall be an equirectangular projection SEI message present in the bitstream that applies to the picture.

When present, a frame packing arrangement SEI message shall be constrained in either of the following ways:

— frame_packing_arrangement_cancel_flag shall be equal to 1.

— All of the following constraints apply:

  — frame_packing_arrangement_type shall be equal to 3, 4, or 5.

  — quincunx_sampling_flag shall be equal to 0.

  — content_interpretation_type shall be equal to 1.

When the video does not cover the entire sphere, for each picture, there shall be a region-wise packing SEI message present in the bitstream that applies to the picture.

When present, the region-wise packing SEI messages shall indicate constraints that comply with the equirectangular projected video scheme type `'erpv'` specified in subclause 7.6.1.3.

10.1.2.3 SEI message related ISO Base Media File Format constraints

This subclause specifies ISOBMFF constraints depending on the presence of omnidirectional video SEI messages of HEVC. This subclause is applicable to the HEVC-based viewport-independent OMAF video profile, the unconstrained HEVC-based viewport-independent OMAF video profile, and the HEVC-based viewport-dependent OMAF video profile as specified subsequently.

The following constraints apply for each picture in the bitstream:

— When the bitstream contains an equirectangular projection SEI message applying to the picture, `ProjectionFormatBox` with `projection_type` equal to 0 shall be present in the sample entry applying to the sample containing the picture.

— When the bitstream contains a cubemap projection SEI message applying to the picture, ProjectionFormatBox with projection_type equal to 1 shall be present in the sample entry applying to the sample containing the picture.

— When the bitstream contains an equirectangular projection SEI message with erp_guard_band_flag equal to 1 applying to the picture, RegionWisePackingBox shall be present in the sample entry applying to the sample containing the picture and shall signal the same information as indicated with the values of erp_guard_band_type, erp_left_guard_band_width, and erp_right_guard_band_width syntax elements of the equirectangular projection SEI message, i.e. the following applies:

  — Let horFact be equal to packed_picture_width / width, where width is the syntax element of the VisualSampleEntry containing the RegionWisePackingBox, and let gbWidth be equal to erp_left_guard_band_width + erp_right_guard_band_width.

  — If SpatiallyPackedStereoFlag is equal to 1, constituent_picture_matching_flag shall be equal to 1. Otherwise, the value of constituent_picture_matching_flag shall be equal to 0.

  — The value of proj_picture_height shall be equal to packed_picture_height.

  — If SideBySideFlag is equal to 1, proj_picture_width shall be equal to packed_picture_width − horFact * 2 * gbWidth. Otherwise, proj_picture_width shall be equal to packed_picture_width − horFact * gbWidth.

  — num_regions shall be equal to 1.

  — guard_band_flag[0] shall be equal to 1.

  — packing_type[0] shall be equal to 0.

  — proj_reg_width[0] shall be equal to proj_picture_width / HorDiv1.

  — proj_reg_height[0] shall be equal to proj_picture_height / VerDiv1.

  — proj_reg_top[0] shall be equal to 0.

  — proj_reg_left[0] shall be equal to 0.

  — transform_type[0] shall be equal to 0.

  — packed_reg_width[0] shall be equal to proj_picture_width / HorDiv1.

  — packed_reg_height[0] shall be equal to proj_picture_height / VerDiv1.

  — packed_reg_top[0] shall be equal to 0.

  — packed_reg_left[0] shall be equal to erp_left_guard_band_width * horFact.

  — left_gb_width[0] shall be equal to erp_left_guard_band_width * horFact.

  — right_gb_width[0] shall be equal to erp_right_guard_band_width * horFact.

  — top_gb_height[0] shall be equal to 0.

  — bottom_gb_height[0] shall be equal to 0.

NOTE When a file writer has no information whether sample values of the guard bands are used in the inter prediction process, gb_not_used_for_pred_flag[0] is suggested to be set equal to 0.

— gb_type[0][j] shall be equal to erp_guard_band_type for each value of j in the range of 0 to 3, inclusive.

— When the bitstream contains a frame packing arrangement SEI message applying to the picture, StereoVideoBox shall be present in the sample entry applying to the sample containing the picture. When StereoVideoBox is present, it shall signal the frame packing format that is included in the frame packing arrangement SEI message(s) in the elementary stream.

— When the bitstream contains a region-wise packing SEI message applying to the picture, RegionWisePackingBox shall be present in the sample entry applying to the sample containing the picture. When present, RegionWisePackingBox shall signal the same information as in the region-wise packing SEI message(s).

10.1.2.4 ISO Base Media File Format constraints

When a track is the only track in a file, compatible_brands containing a brand equal to 'hevi' in FileTypeBox indicates that the track conforms to this media profile. When a file contains multiple tracks, compatible_brands containing a brand equal to 'hevi' in FileTypeBox indicates that at least one of the tracks conforms to this media profile.

compatible_brands containing a brand equal to 'hevi' in TrackTypeBox indicates that the track conforms to this media profile.

A track of this media profile shall be indicated to conform to this media profile through one or both of FileTypeBox and TrackTypeBox.

At least one sample entry type of each sample entry of the track shall be equal to 'resv'.

NOTE 1 'resv' does not have to be the track sample entry type when the track has undergone several transformations. Consequently, this media profile can also be used when the track is protected.

The scheme_type values of SchemeTypeBox in the RestrictedSchemeInfoBox and of all instances of CompatibleSchemeTypeBox in the same RestrictedSchemeInfoBox shall include 'podv' and 'erpv'.

The untransformed sample entry type, derived as specified in ISO/IEC 14496-12, shall be equal to 'hvc1'.

NOTE 2 Consequently, parameter sets are not present inband within samples.

LHEVCConfigurationBox shall not be present in VisualSampleEntry.

HEVCConfigurationBox in VisualSampleEntry shall indicate conformance to the elementary stream constraints specified in subclause 10.1.2.2.

The constraints specified in subclause 10.1.2.3 apply to the track conforming to this media profile, where the bitstream based on which the constraints are derived is reconstructed from the track as specified in subclause 10.1.2.5.

10.1.2.5 File decoding process

The inputs to the file decoding process are

— the track_ID value of the track conforming to this media profile, and

— a file containing at least the track.

An HEVC bitstream is reconstructed from the track with the given `track_ID` value as specified in ISO/IEC 14496-15:2019, Clause 8.

The HEVC bitstream shall conform to the elementary stream constraints specified in subclause 10.1.2.2.

The HEVC bitstream is decoded as specified in Rec. ITU-T H.265 (11/19) | ISO/IEC 23008-2:2020, subclause 8.1.1. The outputs of this process are the same as the outputs of Rec. ITU-T H.265 (11/19) | ISO/IEC 23008-2:2020, subclause 8.1.1. Additionally, for each decoded picture, this process outputs `ProjectionFormatBox`, `StereoVideoBox` (when applicable), `RegionWisePackingBox` (when applicable), and `RotationBox` (when applicable) that provide the input for the semantics of sample locations within the decoded picture as specified in subclause 7.5.1.

10.1.2.6 Expected OMAF player operation

OMAF players conforming to this media profile are expected to process either all referenced SEI messages in subclause 10.1.2.2 or all allowed boxes within the `SchemeInformationBox` for the equirectangular projected video scheme type.

When receiving and playing a DASH presentation and a Representation containing initial viewing orientation metadata, as specified in subclause 7.7.4, is present and associated with a media Representation selected for playing, as specified in subclause 8.2.3, OMAF players are expected to receive and parse the initial viewing orientation metadata Representation and obey it when rendering the media Representation.

10.1.3 HEVC-based viewport-dependent OMAF video profile

10.1.3.1 General

This profile allows unconstrained use of rectangular region-wise packing. With the presence of region-wise packing, the resolution or quality of the omnidirectional video can be emphasized in certain regions, e.g. according to the user's viewing orientation. In addition, the untransformed sample entry type `'hvc2'` is allowed, making it possible to use extractors and get a conforming HEVC bitstream when tile-based streaming is used.

10.1.3.2 Elementary stream constraints

The elementary stream constraints apply to the HEVC bitstream that is reconstructed from a file as specified in subclause 10.1.3.4.

The HEVC bitstream shall comply with the same constraints as the HEVC-based viewport-independent OMAF video profile, with the following exceptions:

— For each picture, there shall be either an equirectangular projection SEI message or a cubemap projection SEI message present in the bitstream that applies to the picture.

— When present, the region-wise packing SEI messages shall indicate constraints that comply with the equirectangular projected video scheme type `'erpv'` specified in subclause 7.6.1.3 or the packed equirectangular or cubemap projected video scheme type `'ercm'` specified in subclause 7.6.1.4.

10.1.3.3 ISO Base Media File Format constraints

When a track is the only track in a file, `compatible_brands` containing a brand equal to `'hevd'` in `FileTypeBox` indicates that the track conforms to this media profile. When a file contains multiple tracks, `compatible_brands` containing a brand equal to `'hevd'` in `FileTypeBox` indicates that at least one of the tracks conforms to this media profile.

`compatible_brands` containing a brand equal to `'hevd'` in `TrackTypeBox` indicates that the track conforms to this media profile.

A track of this media profile shall be indicated to conform to this media profile through one or both of `FileTypeBox` and `TrackTypeBox`.

At least one sample entry type of each sample entry of the track shall be equal to `'resv'`.

NOTE 1 `'resv'` does not have to be the track sample entry type, when the track has undergone several transformations. Consequently, this media profile can also be used when the track is protected.

The `scheme_type` values of `SchemeTypeBox` in the `RestrictedSchemeInfoBox` and of all instances of `CompatibleSchemeTypeBox` in the same `RestrictedSchemeInfoBox` shall include `'podv'` and at least one of `'erpv'` and `'ercm'`.

The untransformed sample entry type, derived as specified in ISO/IEC 14496-12, shall be equal to `'hvc1'` or `'hvc2'`.

When the untransformed sample entry type is `'hvc2'`, the track shall include one or more `'scal'` track references. The referred tracks shall conform to the HEVC-based viewport-independent OMAF video profile or the HEVC-based viewport-dependent OMAF video profile.

For each sub-picture bitstream carried in a referred track, the value of each sample in each decoded sub-picture is identical to the value of the corresponding sample in the decoded reconstructed picture.

NOTE 2 The following two constraints for each sub-picture bitstream carried in a referred track are sufficient to ensure the above constraint to be satisfied, but in some cases, they are more than necessary:

a) It is required that there are no sample values outside the picture that are referenced for inter prediction.

b) For prediction units located on the right-side picture boundary except the last one at the bottom-right corner of the picture, the following applies when CuPredMode[xPb][yPb] is equal to MODE_INTER, where (xPb, yPb) specifies the top-left sample of the corresponding luma prediction block relative to the top-left sample of the current picture:

1) With the number of spatial merging candidates numSpatialMergeCand derived as follows:

```
numSpatialMergeCand = $\text{availableFlagA}_0$ + $\text{availableFlagA}_1$ +
                      $\text{availableFlagB}_0$ + $\text{availableFlagB}_1$ + $\text{availableFlagB}_2$
```

where availableFlagA_0, availableFlagA_1, availableFlagB_0, availableFlagB_1, and availableFlagB_2 are the output of the derivation process for spatial merging candidates specified in Rec. ITU-T H.265 (11/19) | ISO/IEC 23008-2:2020, subclause 8.5.3.2.3, the following applies:

i) If numSpatialMergeCand is equal to 0, merge_flag[xPb][yPb] is required to be equal to 0.

ii) Otherwise (numSpatialMergeCand is greater than 0), merge_idx[xPb][yPb] is required to be in the range of 0 to numSpatialMergeCand − 1, inclusive.

2) With the number of spatial motion vector predictor candidates numSpatialMvpCand derived as follows:

```
if ( availableFlagLXA )
    numSpatialMvpCand = availableFlagLXA +
            ( ( mvLXA != mvLXB ) ? availableFlagLXB : 0 )
else
    numSpatialMvpCand = availableFlagLXB
```

where availableFlagLXA, availableFlagLXB, mvLXA, and mvLXB are the output of the derivation process for motion vector predictor candidates from neighbouring prediction unit partitions specified in Rec. ITU-T H.265 (11/19) | ISO/IEC 23008-2:2020, subclause 8.5.3.2.7, the following applies:

iii) If numSpatialMvpCand is equal to 0, mvp_l0_flag[xPb][yPb] and mvp_l1_flag[xPb][yPb] are both required to be equal to 1.

iv) Otherwise (numSpatialMvpCand is greater than 0), mvp_l0_flag[xPb][yPb] and mvp_l1_flag[xPb][yPb] are both required to be in the range of 0 to numSpatialMvpCand − 1, inclusive.

Note that the first constraint above restricts that motion vectors point to full-sample locations inside the picture and to fractional-sample locations that require only full-sample locations inside the picture for interpolation. Note that in the above constraints and the previous sentence, a sub-picture becomes a picture within the context of one sub-picture bitstream. The second constraint restricts that, when the sub-picture bitstream of the referred track together with other sub-picture bitstreams carried in other referred tracks are reconstructed into one conforming bitstream, in decoding of the entire reconstructed bitstream, for blocks of the sub-picture of this sub-picture bitstream, there won't be motion vector candidates for temporal motion vector prediction derived from blocks outside the "sub-picture".

NOTE 3 Besides the above constraint, the following constraints are also expected to be satisfied:

a) When multiple tiles are present in the bitstream resolved from an extractor track, the value of entropy_coding_sync_enabled_flag in the active PPS is required be equal to 0 for each sub-picture bitstream carried in a referred track.

b) The referred tracks are required to contain the same number of media samples.

c) The samples with the same sample number across the referred tracks are required to have the same presentation time.

d) The pictures carried in the samples with the same sample number across the referred tracks are required shall have the same value of picture order count, i.e. PicOrderCntVal.

`LHEVCConfigurationBox` shall not be present in `VisualSampleEntry`.

`HEVCConfigurationBox` in `VisualSampleEntry` shall indicate conformance to the elementary stream constraints specified in subclause 10.1.3.2.

The `track_not_intended_for_presentation_alone` flag of the `TrackHeaderBox` may be used to indicate that a track is not intended to be presented alone.

The constraints specified in subclause 10.1.2.3 apply to the track conforming to this media profile, where the bitstream based on which the constraints are derived is reconstructed from the track as specified in subclause 10.1.3.4.

10.1.3.4 File decoding process

The inputs to the file decoding process are

— the `track_ID` value of the track conforming to this media profile, and

— a file containing at least the track with that `track_ID` value and all tracks that the track directly or indirectly depends on.

An HEVC bitstream is reconstructed as follows:

— If the untransformed sample entry type of the track with the given `track_ID` is equal to `'hvc1'`, an HEVC bitstream is generated from the track with the given `track_ID` value as specified in ISO/IEC 14496-15:2019, Clause 8.

— Otherwise (the untransformed sample entry type of the track with the given `track_ID` is equal to `'hvc2'`), an HEVC bitstream is generated from the track with the given `track_ID` value as specified in ISO/IEC 14496-15:2019, Clause 9 and Annex A.

The HEVC bitstream shall conform to the elementary stream constraints specified in subclause 10.1.3.2.

The HEVC bitstream is decoded as specified in Rec. ITU-T H.265 (11/19) | ISO/IEC 23008-2:2020, subclause 8.1.1. The outputs of this process are the same as the outputs of Rec. ITU-T H.265 (11/19) | ISO/IEC 23008-2:2020, subclause 8.1.1. Additionally, for each decoded picture, this process outputs `ProjectionFormatBox`, `StereoVideoBox` (when applicable), `RegionWisePackingBox` (when applicable), and `RotationBox` (when applicable) that provide the input for the semantics of sample locations within the decoded picture as specified in subclause 7.5.1.

10.1.3.5 Expected OMAF player operation

OMAF players conforming to this media profile are expected to process either all referenced SEI messages in subclause 10.1.3.2 or all allowed boxes within the `SchemeInformationBox` for the `'erpv'` and `'ercm'` scheme types.

When receiving and playing a DASH presentation and a Representation containing initial viewing orientation metadata, as specified in subclause 7.7.4, is present and associated with a media Representation selected for playing, as specified in subclause 8.2.3, OMAF players are expected to receive and parse the initial viewing orientation metadata Representation and obey it when rendering the media Representation.

A player conforming to the HEVC-based viewport-dependent OMAF video profile is expected to:

— Parse both `SphereRegionQualityRankingBox` and `2DRegionQualityRankingBox`, when present, of tracks present in a file and select the track that matches user's viewing orientation. The player should use `2DRegionQualityRankingBox` together with `RegionWisePackingBox`, when present, to conclude which sphere regions the indicated quality ranking 2D regions correspond to.

— Parse spherical region-wise quality ranking (SRQR) descriptors, when present, and select the Adaptation Sets and Representations that match the user's viewing orientation:

  — When Preselections are applied in the MPD, the player is expected to select a Main Adaptation Set based on the SRQR descriptors.

  — When `@dependencyId` is applied in the MPD, the player is expected to select the dependent Representation based on the SRQR descriptors.

10.1.4 AVC-based viewport-dependent OMAF video profile

10.1.4.1 General

This media profile allows unconstrained use of rectangular region-wise packing with AVC. With the presence of region-wise packing, the resolution of the omnidirectional video can be emphasized in certain regions, e.g. according to the user's viewing orientation. In addition, the untransformed sample entry types `'avc2'` and `'avc4'` are allowed, making it possible to use extractors and get a conforming AVC bitstream when slice-based streaming is used.

10.1.4.2 Elementary stream constraints

The elementary stream constraints apply to the AVC bitstream that is reconstructed from a file as specified in subclause 10.1.4.4.

The bitstream shall comply with AVC Progressive High profile, Level 5.1.

10.1.4.3 ISO Base Media File Format constraints

When a track is the only track in a file, `compatible_brands` containing a brand equal to `'avde'` in `FileTypeBox` indicates that the track conforms to this media profile. When a file contains multiple tracks, `compatible_brands` containing a brand equal to `'avde'` in `FileTypeBox` indicates that at least one of the tracks conforms to this media profile.

`compatible_brands` containing a brand equal to `'avde'` in `TrackTypeBox` indicates that the track conforms to this media profile.

A track of this media profile shall be indicated to conform to this media profile through one or both of `FileTypeBox` and `TrackTypeBox`.

At least one sample entry type of each sample entry of the track shall be equal to `'resv'`.

NOTE `'resv'` does not have to be the track sample entry type, when the track has undergone several transformations. Consequently, this media profile can also be used when the track is protected.

The `scheme_type` values of `SchemeTypeBox` in the `RestrictedSchemeInfoBox` and of all instances of `CompatibleSchemeTypeBox` in the same `RestrictedSchemeInfoBox` shall include `'podv'` and at least one of `'erpv'` and `'ercm'`.

The untransformed sample entry type, derived as specified in ISO/IEC 14496-12, shall be equal to `'avc1'`, `'avc2'`, `'avc3'`, or `'avc4'`.

When the untransformed sample entry type is `'avc2'` or `'avc4'`, the track shall include one or more `'scal'` track references.

`AVCConfigurationBox` in `VisualSampleEntry` shall indicate conformance to the elementary stream constraints specified in subclause 10.1.4.2.

10.1.4.4 File decoding process

The inputs to the file decoding process are

— the `track_ID` value of the track conforming to this media profile, and

— a file containing at least the track with that `track_ID` value and all tracks that the track directly or indirectly depends on.

An AVC bitstream is reconstructed as follows:

— If the untransformed sample entry type of the track with the given `track_ID` is equal to `'avc1'` or `'avc3'`, an AVC bitstream is generated from the track with the given `track_ID` value as specified in ISO/IEC 14496-15:2019, Clause 5.

— Otherwise (the untransformed sample entry type of the track with the given `track_ID` is equal to `'avc2'` or `'avc4'`), an AVC bitstream is generated from the track with the given `track_ID` value as specified in ISO/IEC 14496-15:2019, Clause 5 and Annex A.

The AVC bitstream shall conform to the elementary stream constraints specified in subclause 10.1.4.2.

The AVC bitstream is decoded as specified in Rec. ITU-T H.264 (06/19) | ISO/IEC 14496-10:2014, Clause 8. The outputs of this process are the same as the outputs of Rec. ITU-T H.264 (06/19) | ISO/IEC 14496-10:2014, Clause 8. Additionally, for each decoded picture, this process outputs `ProjectionFormatBox`, `StereoVideoBox` (when applicable), `RegionWisePackingBox` (when applicable), and `RotationBox` (when applicable) that provide the input for the semantics of sample locations within the decoded picture as specified in subclause 7.5.1.

10.1.4.5 Expected OMAF player operation

OMAF players conforming to this media profile are expected to process all allowed boxes within the `SchemeInformationBox` for the `'erpv'` and `'ercm'` scheme types.

When receiving and playing a DASH presentation and a Representation containing initial viewing orientation metadata, as specified in subclause 7.7.4, is present and associated with a media Representation selected for playing, as specified in subclause 8.2.3, OMAF players are expected to receive and parse the initial viewing orientation metadata Representation and obey it when rendering the media Representation.

A player conforming to the AVC-based viewport-dependent OMAF video profile is expected to:

— Parse both `SphereRegionQualityRankingBox` and `2DRegionQualityRankingBox`, when present, of tracks present in a file and select the track that matches user's viewing orientation. The player should use `2DRegionQualityRankingBox` together with `RegionWisePackingBox`, when present, to conclude which sphere regions the indicated quality ranking 2D regions correspond to.

— Parse spherical region-wise quality ranking (SRQR) descriptors, when present, and select the Adaptation Sets and Representations that match the user's viewing orientation:

  — When Preselections are applied in the MPD, the player is expected to select a Main Adaptation Set based on the SRQR descriptors.

  — When `@dependencyId` is applied in the MPD, the player is expected to select the dependent Representation based on the SRQR descriptors.

10.1.5 Unconstrained HEVC-based viewport-independent OMAF video profile

10.1.5.1 General

This profile provides the same set of features as the HEVC-based viewport-independent OMAF video profile. The only difference is that there is no constraint on the HEVC bitstream level. Therefore, service providers can use this profile with a variety of resolutions and not only with resolutions conforming to HEVC level 5.1. This profile enables, for instance, content being prepared and decoded at a resolution conforming to HEVC level 6.1, such as 8192×4096.

The content author signals the level of the bitstream by setting the appropriate value in the general_level_idc of the HEVC bitstream. This value of general_level_idc is additionally available in the ISOBMFF encapsulation of the bitstream in the `HEVCDecoderConfigurationRecord` structure of `HEVCConfigurationBox` contained in the HEVC sample entry definition specified in ISO/IEC 14496-15. Lastly, the `@codecs` attribute in the DASH MPD exposes in a single string the level indication among other information.

10.1.5.2 Elementary stream constraints

The elementary stream constraints apply to the HEVC bitstream that is reconstructed from a file as specified in subclause 10.1.5.4.

The bitstream shall comply with HEVC Main 10 profile, Main tier and any level.

The elementary stream shall further comply with the constraint defined in subclause 10.1.2.2 except for the bitstream constraint on profile, tier and level which are defined in this section.

10.1.5.3 ISO Base Media File Format constraints

When a track is the only track in a file, `compatible_brands` containing a brand equal to `'uhvi'` in `FileTypeBox` indicates that the track conforms to this media profile. When a file contains multiple tracks, `compatible_brands` containing a brand equal to `'uhvi'` in `FileTypeBox` indicates that at least one of the tracks conforms to this media profile.

`compatible_brands` containing a brand equal to `'uhvi'` in `TrackTypeBox` indicates that the track conforms to this media profile.

A track of this media profile shall be indicated to conform to this media profile through one or both of `FileTypeBox` and `TrackTypeBox`.

At least one sample entry type of each sample entry of the track shall be equal to 'resv'.

NOTE 1 'resv' does not have to be the track sample entry type when the track has undergone several transformations. Consequently, this media profile can also be used when the track is protected.

The scheme_type values of SchemeTypeBox in the RestrictedSchemeInfoBox and of all instances of CompatibleSchemeTypeBox in the same RestrictedSchemeInfoBox shall include 'podv' and 'erpv'.

The untransformed sample entry type, derived as specified in ISO/IEC 14496-12, shall be equal to 'hvc1'.

NOTE 2 Consequently, parameter sets are not present inband within samples.

LHEVCConfigurationBox shall not be present in VisualSampleEntry.

HEVCConfigurationBox in VisualSampleEntry shall indicate conformance to the elementary stream constraints specified in subclause 10.1.5.2.

The constraints specified in subclause 10.1.2.3 apply to the track conforming to this media profile, where the bitstream based on which the constraints are derived is reconstructed from the track as specified in subclause 10.1.5.4.

10.1.5.4 File decoding process

The inputs to the file decoding process are

— the track_ID value of the track conforming to this media profile, and

— a file containing at least the track.

An HEVC bitstream is reconstructed from the track with the given track_ID value as specified in ISO/IEC 14496-15:2019, Clause 8.

The HEVC bitstream shall conform to the elementary stream constraints specified in subclause 10.1.5.2.

The HEVC bitstream is decoded as specified in Rec. ITU-T H.265 (11/19) | ISO/IEC 23008-2:2020, subclause 8.1.1. The outputs of this process are the same as the outputs of Rec. ITU-T H.265 (11/19) | ISO/IEC 23008-2:2020, subclause 8.1.1. Additionally, for each decoded picture, this process outputs ProjectionFormatBox, StereoVideoBox (when applicable), RegionWisePackingBox (when applicable), and RotationBox (when applicable) that provide the input for the semantics of sample locations within the decoded picture as specified in subclause 7.5.1.

10.1.5.5 Expected OMAF player operation

OMAF players conforming to this media profile are expected to operate as explained in subclause 10.1.2.6.

10.1.6 Advanced tiling OMAF video profile

10.1.6.1 General

This subclause captures the spirit of the profile in a human-friendly manner. The subsequent subclauses specify the constraints to enable the interoperability point.

The profile is based on the tile-base streaming with late binding as described in subclause 4.5 and on the use of segment formats specified in subclause 8.6. In addition, the profile relies on a number of principles, which are described below:

- The content is represented using a 3D mesh that is constructed either from parallelograms or from sphere regions of shape type 1.
- Multiple resolutions of the content may be created to provide different quality levels, and to provide the capability to simultaneously decode different areas of the sphere at different qualities and resolutions.
- Each representation (resolution) of the content is divided into a number of tile sequences of equal size.
  - The tile sequences have a configurable guard band to allow rendering in the client without seams.
- The tile sequences are individually encoded using HEVC, with a number of restrictions, mostly motion vector constraints and limiting the configuration to a single tile per slice.

10.1.6.2 Elementary stream constraints

10.1.6.2.1 Bitstream constraints at the decoder

The elementary stream constraints in this subclause apply to the bitstream that is reconstructed as specified in subclause 10.1.5.4 after bitstream rewriting.

The bitstream shall comply with HEVC Main 10 profile, Main tier. The present Profile definition does not specify a Level constraint so as to allow the Profile to be used with HEVC Main 10 Profile at various Levels.

All pictures shall be encoded as coded frames, and shall not be encoded as coded fields.

There shall be exactly one slice per tile and exactly one tile per slice.

10.1.6.2.2 Encoding constraints

In order to the decoding operations to work, the following constraints need to be observed at the encoder:

- The tiling grid shall be constant for each entire coded video sequence.
- Each tile shall be motion-constrained as specified in the semantics of the temporal motion-constrained tile sets SEI message of Rec. ITU-T H.265 | ISO/IEC 23008-2.

All the active SPSs of the bitstream shall be constrained as follows:

- general_progressive_source_flag shall be equal to 1.
- general_frame_only_constraint_flag shall be equal to 1.
- general_interlaced_source_flag shall be equal to 0.

When encoding multiple quality levels, or when using individual encoders for each tile, the following constraints need to be observed for all time-aligned tiles and all levels that need to be able to be merged into a single compliant bitstream by the client:

- init_qp_minus26 in the PPS shall be set to the same value across tiles and levels.
- The reference picture set (RPS) shall be the same across tiles and levels.

10.1.6.3 ISO base media file format constraints

10.1.6.3.1 Stream header

A Stream Header shall conform to subclause 8.6.1.2 and the following constraints:

— When a track is the only track in a file, `compatible_brands` containing a brand equal to `'adti'` in `FileTypeBox` indicates that the track conforms to this media profile. When a file contains multiple tracks, `compatible_brands` containing a brand equal to `'adti'` in `FileTypeBox` indicates that at least one of the tracks conforms to this media profile.

— A track of this media profile shall be indicated to conform to this media profile through one or both of `FileTypeBox` and `TrackTypeBox`.

NOTE The ISOBMFF boxes and nesting, optionality and ordinality of the Stream Header are described in Table 50.

Table 50 — Stream Header for the advanced tiling OMAF video profile

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	Format Req.	Specification	Constraints	Description
`ftyp`									1	ISO/IEC 14496-12:2020, subclause 4.3	This subclause	
Either `moov` or `!mov`									1	ISO/IEC 14496-12:2020, subclauses 8.2.1 and 8.19.6		
	`mvhd`								1	ISO/IEC 14496-12:2020, subclause 8.2.2	ISO/IEC 23000-19:2020, subclause 7.5.1	
	`mvex`								1	ISO/IEC 14496-12:2020, subclause 8.8.1		
		`mehd`							1	ISO/IEC 14496-12:2020, subclause 8.8.2	ISO/IEC 23000-19:2020, subclause 7.5.1	
		`trex`							1	ISO/IEC 14496-12:2020, subclause 8.8.3	ISO/IEC 23000-19:2020, subclause 7.5.14	

10.1.6.3.2 Track box format for OMAF base track and OMAF tile tracks

Each of the track boxes of the OMAF base track and the OMAF tile tracks is constrained as follows:

— The `TrackBox` of the OMAF base track shall conform to subclause 8.6.1.3.

— The `TrackBox` of each OMAF tile track shall conform to subclause 8.6.1.4.

— At least one sample entry type of each sample entry of the track shall be equal to `'resv'`.

NOTE 1 `'resv'` does not have to be the track sample entry type, when the track has undergone several transformations. Consequently, this media profile can also be used when the track is protected.

- The scheme_type values of SchemeTypeBox in the RestrictedSchemeInfoBox and of all instances of CompatibleSchemeTypeBox in the same RestrictedSchemeInfoBox shall include 'modv'and 'meov'.

- The untransformed sample entry type of the OMAF base track shall be equal to 'hvc2'. The OMAF base track shall conform to all constraints of the 'hvc2' sample entry type, as specified in ISO/IEC 14496-15.

- The untransformed sample entry type of each OMAF tile track shall be equal to 'hvt3'. Each OMAF tile track shall conform to all constraints of the 'hvt3' sample entry type, as specified in ISO/IEC 14496-15.

- The OMAF base track shall contain a MeshBox.

- The OMAF base track may contain a StereoVideoBox but the mapping between a rectangular region and its mapping on the left or right eye is signalled through the TileMeshGroupEntry.

- The OMAF base track shall contain 'sabt' track references to all OMAF tile tracks associated with this OMAF base track or to 'alte' track groups, each containing collocated OMAF tile tracks of the same width and height and associated with this OMAF base track.

- Each OMAF tile track shall contain 'tbas' track reference to the associated OMAF base track.

- The PPS carried in the hvc2 structure shall have tiles_enabled set to true, with num_tile_columns_minus1 and num_tile_rows_minus1 set to match the number of 'sabt' track references associated with this particular sub-picture collection track. The SPS shall be set to the full dimensions of this layer (i.e. num_tile_columns * tile_width and num_tile_rows * tile_height).

- Each OMAF tile track shall contain a tile mesh sample group as specified in subclause 7.16.2.

- Each OMAF tile track shall contain a 'trif' sample group as specified in ISO/IEC 14496-15.

NOTE 2 The ISOBMFF boxes and nesting, optionality and ordinality of the TrackBox for the OMAF base track are described in Table 51.

Table 51 — Track box for OMAF base track for the advanced tiling OMAF video profile

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	NL 9	Format Req.	Specification	Constraints	Description
	`trak`									1	ISO/IEC 14496-12:2020, subclause 8.3.1		
		`tkhd`								1	ISO/IEC 14496-12:2020, subclause 8.3.2	ISO/IEC 23000-19:2020, subclause 7.5.4 except that there are no constraints beyond those in ISO/IEC 14496-12 on the `width` and `height` fields and with the additional constraint that the field `matrix` shall always be set to the default values as defined in ISO/IEC 14496-12.	
		`tref`								1	ISO/IEC 14496-12:2020, subclause 8.3.3	This subclause	Track references to OMAF tile tracks.
		`mdia`								1	ISO/IEC 14496-12:2020, subclause 8.4		
			`mdhd`							1	ISO/IEC 14496-12:2020, subclause 8.4.2	ISO/IEC 23000-19:2020, subclause 7.5.5	
			`hldr`							1	ISO/IEC 14496-12:2020, subclause 8.4.3		
			`minf`							1	ISO/IEC 14496-12:2020, subclause 8.4.4		
				`vmhd`						1	ISO/IEC 14496-12:2020, subclause 12.1.2	ISO/IEC 23000-19:2020, subclause 7.5.6	
				`dinf`						1	ISO/IEC 14496-12:2020, subclause 8.7.1		
					`dref`					1	ISO/IEC 14496-12:2020, subclause 8.7.2		
						`snim`				1	ISO/IEC 14496-12:2020, subclause 8.7.2.2		

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	NL 9	Format Req.	Specification	Constraints	Description
				stbl						1	ISO/IEC 14496-12:2020, subclause 8.5	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stsd					1	ISO/IEC 14496-12:2020, subclause 8.5.2	ISO/IEC 23000-19:2020, subclause 7.5.10	
						resv				1			
						rinf				1	ISO/IEC 14496-12:2020, subclause 8.15.3	This subclause	
							hvc2			1	ISO/IEC 14496-15:2019, subclause 8.4	This subclause	
							schm			1	ISO/IEC 14496-12:2020, subclause 8.12.5		
							schi			1	ISO/IEC 14496-12:2020, subclause 8.12.6		
								movd		1	Subclause 7.6.7	Subclause 7.6.7	
									mesh	1	Subclause 7.6.8	Subclause 7.6.8	
									rotn	0/1	Subclause 7.6.5	Subclause 7.6.5	
					stts					1	ISO/IEC 14496-12:2020, subclause 8.6.1.2	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stsc					1	ISO/IEC 14496-12:2020, subclause 8.7.4	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stco					1	ISO/IEC 14496-12:2020, subclause 8.7.5	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stsz					1	ISO/IEC 14496-12:2020, subclause 8.7.3	ISO/IEC 23000-19:2020, subclause 7.5.12	

NOTE 3 The ISOBMFF boxes and nesting, optionality and ordinality of the `TrackBox` for each OMAF tile track referenced by the OMAF base track are described in Table 52.

Table 52 — Track box for OMAF tile track for the advanced tiling OMAF video profile

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	NL9	Format Req.	Specification	Constraints	Description
	trak									+	ISO/IEC 14496-12:2020, subclause 8.3.1		
		tkhd								1	ISO/IEC 14496-12:2020, subclause 8.3.2	ISO/IEC 23000-19:2020, subclause 7.5.4 except that there are no constraints beyond those in ISO/IEC 14496-12 on the `width` and `height` fields and with the additional constraint that the field `matrix` shall always be set to the default values as defined in ISO/IEC 14496-12.	
		mdia								1	ISO/IEC 14496-12:2020, subclause 8.4		
			mdhd							1	ISO/IEC 14496-12:2020, subclause 8.4.2	ISO/IEC 23000-19:2020, subclause 7.5.5	
			hldr							1	ISO/IEC 14496-12:2020, subclause 8.4.3		
			minf							1	ISO/IEC 14496-12:2020, subclause 8.4.4		
				vmhd						1	ISO/IEC 14496-12:2020, subclause 12.1.2	ISO/IEC 23000-19:2020, subclause 7.5.6	
				dinf						1	ISO/IEC 14496-12:2020, subclause 8.7.1		
					dref					1	ISO/IEC 14496-12:2020, subclause 8.7.2		
						snim				1	ISO/IEC 14496-12:2020, subclause 8.7.2.2		
				stbl						1	ISO/IEC 14496-12:2020, subclause 8.5	ISO/IEC 23000-19:2020, subclause 7.5.12	

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	NL9	Format Req.	Specification	Constraints	Description
					`stsd`					1	ISO/IEC 14496-12:2020, subclause 8.5.2	ISO/IEC 23000-19:2020, subclause 7.5.10	
						`resv`				1			
						`rinf`				1	ISO/IEC 14496-12:2020, subclause 8.15.3	This subclause	
							`hvt3`			1	ISO/IEC 14496-15:2019, subclause 10.5	This subclause	
							`schm`			1	ISO/IEC 14496-12:2020, subclause 8.12.5		
							`schi`			1	ISO/IEC 14496-12:2020, subclause 8.12.6		
					`sgpd`					1	ISO/IEC 14496-12:2020, subclause 8.9.3		
						`tmsh`				1	This document	Subclause 7.16.2	
					`stts`					1	ISO/IEC 14496-12:2020, subclause 8.6.1.2	ISO/IEC 23000-19:2020, subclause 7.5.12	
					`stsc`					1	ISO/IEC 14496-12:2020, subclause 8.7.4	ISO/IEC 23000-19:2020, subclause 7.5.12	
					`stss`					0/1	ISO/IEC 14496-12:2020, subclause 8.6.2	ISO/IEC 23000-19:2020, subclause 7.5.12	
					`stco`					1	ISO/IEC 14496-12:2020, subclause 8.7.5	ISO/IEC 23000-19:2020, subclause 7.5.12	
					`stsz`					1	ISO/IEC 14496-12:2020, subclause 8.7.3	ISO/IEC 23000-19:2020, subclause 7.5.12	

10.1.6.4 Bitstream rewriting

10.1.6.4.1 General

Figure 30 illustrates the organisation of video data into slices and their respective preceding slice segment header stored in a sequence of Network Abstraction Layer (NAL) units constituting an HEVC bitstream.

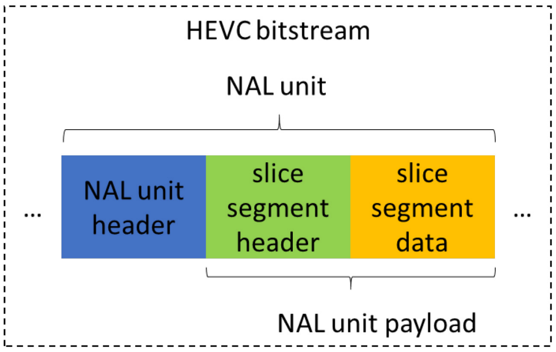


Figure 30 — Video data in HEVC bitstream

One element that needs to be taken into account while rewriting any part of the HEVC bitstream is start code emulation prevention bytes (see Rec. ITU-T H.265 (11/19) | ISO/IEC 23008-2:2020, subclause 7.4.2). Insertion, removal or changing of any fields in the HEVC bitstream can impact whether emulation prevention bytes need to be added or removed as well. One method for making sure the resulting bitstream is compliant is to completely remove any emulation prevention bytes from the NAL unit before starting the rewriting procedure, and to re-add them after the rewriting of the NAL unit is complete.

It is up to an OMAF player to determine, at runtime, not just which slices to download and decode, but also where to place them in the decoded picture. This can be controlled by rewriting the slice_segment_address field in the HEVC slice segment header. However, in order to produce a compliant HEVC bitstream, and not break any inter-picture dependencies, it is important that a sequence of slices that was collocated in the original encoded bitstream uses the same slice_segment_address in the rewritten bitstream until it no longer needs to be decoded/displayed. In other words, if a particular slice becomes covered by the viewport at frame N, and the player decides to assign a certain slice_segment_address value to it, the slices that are collocated in the original encoded bitstream with that particular slice should keep that slice_segment_address value until the viewport no longer covers that particular slice at frame M, regardless of the slice perhaps residing in different parts of the viewport over time.

The following subclauses assume that during encoding each tile is packaged into exactly one slice, and each slice contains exactly one tile. While there is nothing preventing this media profile to be used in combination with multiple tiles per slice (or multiples slices per tile), this makes the rewriting procedure slightly more complex and is therefore not included in this subclause 10.1.6.4. For the same reason, the text in the following subclauses assumes that there are no dependent slices present in the bitstream.

10.1.6.4.2 Rewriting parameter sets

The Video Parameter Set (VPS), Sequence Parameter Set (SPS) and Picture Parameter Set (PPS) carried in the Initialization Segment need to be updated to signal the correct resolution (SPS) and number of tile rows/columns (PPS) that will be decoded. Note that in contrast to traditional video delivery and the HEVC-based viewport-dependent OMAF video profile, with late binding it is up to the client to determine the optimal values for these parameters at runtime. Elements that the client can take into account when determining these values include:

— content characteristics,

— device characteristics,

— display field-of-view,

— display resolution, and

— current zoom-level.

Depending on the implementation, the client can decide to set these values once (i.e. by dimensioning for the highest expected number of simultaneous tiles during the session) or dynamically, adjusting them as the number of tiles that need to be decoded changes over time.

a) In the VPS, the content of profile_tier_level( ) are set to a value that matches the profile that is being targeted (i.e. based on the amount of tiles being decoded and the resulting resolution).

Video_parameter_set_rbsp( ) {	**Descriptor**
…	
profile_tier_level(1, vps_max_sub_layers_minus1)	
…	
}	

profile_tier_level(profilePresentFlag, maxNumSubLayersMinus1) {	**Descriptor**
…	
general_level_idc	u(8)
…	
for(I = 0; i < maxNumSubLayersMinus1; i++) {	
if(sub_layer_level_present_flag[i])	
sub_layer_level_idc*[i]*	u(8)
}	
}	

b) In the SPS, pic_width_in_luma_samples and pic_height_in_luma_samples are recalculated and set according to the number of tile rows and columns being decoded along with the tile size.

seq_parameter_set_rbsp( ) {	**Descriptor**
…	
pic_width_in_luma_samples	ue(v)
pic_height_in_luma_samples	ue(v)
…	
}	

c) In the PPS, tiles_enabled_flag is set to 1, and num_tile_columns_minus1 and num_tile_rows_minus1 are set based on the maximum number of tile rows and columns that the client wishes to decode. In cases where a uniform tile size is used, uniform_spacing_flag is set to 1. Finally, in most cases loop_filter_across_tiles_enabled_flag and pps_loop_filter_across_slices_enabled_flag are set to 0.

pic_parameter_set_rbsp( ) {	**Descriptor**
...	
tiles_enabled_flag	u(1)
...	
if(tiles_enabled_flag) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width_minus1*[i]*	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height_minus1*[i]*	ue(v)
}	
loop_filter_across_tiles_enabled_flag	u(1)
}	
pps_loop_filter_across_slices_enabled_flag	u(1)
...	
}	

10.1.6.4.3 Rewriting slice segment header and possibly NAL unit header for each VCL NAL unit

While the exact fields that need to be rewritten depend on the encoding parameters used, at the absolute minimum the first_slice_segment_in_pic_flag and slice_segment_address, which indicate the spatial position of a tile within a decoded frame, are rewritten. Given that the slice_segment_address is variable-length encoded, with the length depending on the configured decoder dimensions (as signalled in the SPS), it can happen that during the rewriting process the length of the slice_segment_address changes. If that is the case, it is likely the byte_alignment structure at the end of the slice segment header needs to be rewritten.

slice_segment_header( ) {	**Descriptor**
first_slice_segment_in_pic_flag	u(1)
...	
if(!first_slice_segment_in_pic_flag) {	
...	
slice_segment_address	u(v)
}	
...	
byte_alignment( )	
}	

To assist in the process of rewriting the slice segment header, the Slice Segment Header Information NAL-unit-like structure as specified by the `'hvt3'` samples provide relevant information. Figure 31 represents the organisation of a Tile Data Segment containing Slice Segment Header Information NAL-unit-like preceding HEVC VCL NAL units.

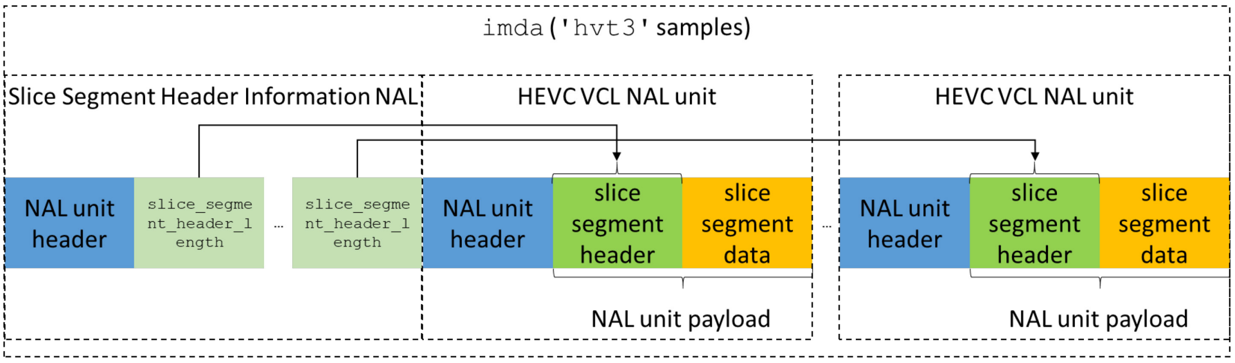


Figure 31 — Slice Segment Header Information NAL-unit-like structure in `'hvt3'` tracks

The client may execute the following steps to locate the first_slice_segment_in_pic_flag, the slice_segment_address and the byte_alignment( ) fields in each VCL NAL unit in order to rewrite the slice segment header:

— Based on the information in the `TrackRunBox` (`'trun'`), the client can determine the location of a particular sample (VCL NAL unit) in the `IdentifiedMediaDataBox` (`'imda'`). Given that the NAL unit header has a fixed length in HEVC, this directly leads a client to the position of the first_slice_segment_in_pic_flag field.

— Once the location of the first_slice_segment_in_pic_flag field in the bitstream is known, the location of the slice_segment_address field can be calculated by simply parsing the slice segment header field in between. Given that the number of fields that need to be parsed is limited, this is not likely to result in a significant impact on performance.

— As specified in Rec. ITU-T H.265 (11/19) | ISO/IEC 23008-2:2020, subclause 7.4.7.1, the value of new slice_segment_address determines the location of the slice in the decoded picture and is expressed in coding tree block raster scan of a picture. If the picture resolution does not change in the rewriting process, the length of this field remains the same. In addition, if the tiling grid does not change in the rewriting, the list of slice_segment_address values for a picture does not change.

— If, based on the desired decoder dimensions and the rewritten pic_width_in_luma_samples and pic_height_in_luma_samples, the length in bits of the slice_segment_address has changed, the client can need to rewrite the byte_alignment( ) structure at the end of the slice segment header accordingly. To find the location of this structure, a client can use the information embedded in the Slice Segment Header Information NAL-unit-like structure (see Figure 31), to find the length of the slice segment header, and through it directly find the position of the byte_alignment( ) structure.

10.1.6.5 Expected OMAF player operation

When determining which OMAF tile tracks are received, `TileMeshGroupEntry` can be used to identify location of individual OMAF tile tracks, and whether they are collocated.

It is assumed that OMAF tile tracks representing an entire low-quality omnidirectional picture are fetched. Once decoded, the low-quality omnidirectional picture is rendered onto a rendering mesh and subsequently the high-quality HEVC tiles are superimposed on top of the low quality omnidirectional picture on the rendering mesh.

10.1.7 Simple tiling OMAF video profile

10.1.7.1 General

This subclause captures the spirit of the profile in a human-friendly manner. The subsequent subclauses specify the constraints to enable the interoperability point.

The profile enables tile-base streaming with free-viewport author-driven binding as described in subclause 4.5 and is based on the use of segment formats specified in subclause 8.6. In addition, the profile relies on a number of principles, which are described below:

— The content is represented by using the equirectangular or cubemap projection.

— The profile supports providing multiple resolutions of the content at different quality levels.

— The content is encoded as encoded tile sequences, which represent areas of the sphere at different qualities and resolutions.

— Encoded tile sequences are made available as OMAF tile tracks.

— One or more OMAF base tracks are authored, e.g. for different decoding capabilities, and made available for streaming.

  — An OMAF base track specifies instructions to rewrite the content of the OMAF tile tracks referenced by the OMAF base track into a conforming HEVC bitstream.

  — Groups of OMAF tile tracks that are alternatives to be referenced by the OMAF base track are indicated in the Stream Header (i.e. in the Initialization Segment used in DASH).

— An OMAF player determines which OMAF base track is consumed e.g. based on available decoding resources.

— The OMAF player determines which OMAF tile track among each referenced group of alternative OMAF tile tracks is retrieved. The OMAF player can retrieve a full set of low-resolution encoded tile sequences to show the full representation of the content at a lower quality and a subset of high-resolution encoded tile sequences that together cover the viewport.

— The OMAF player follows the instructions of the OMAF base track to rewrite the content of the selected OMAF tile tracks into an HEVC bitstream.

— It is possible to use the simple tiling OMAF video profile and overlay video jointly in either of the following ways:

  — One or more overlay video tracks are present separately from any OMAF base track representing the background video. This option requires multiple decoder instances; one for the selected OMAF base track and one per each overlay video track.

  — An OMAF base track references OMAF tile tracks, each containing either a portion of background video or overlay video. This option requires only a single video decoder instance in the OMAF player. Regardless of whether OMAF tile tracks contain background video or overlay video, they are treated similarly in all OMAF player operations except content selection and rendering.

  NOTE When none of the OMAF tile tracks contains overlays that are essential for displaying, content authors are advised to create a second OMAF base track that does not reference any OMAF tile tracks containing overlays. Thus, those OMAF players that do not support overlays can select the second OMAF base track for their use.

10.1.7.2 Elementary stream constraints

The elementary stream constraints apply to the HEVC bitstream that is reconstructed from a file as specified in subclause 10.1.7.3.3.

The bitstream shall comply with HEVC Main 10 profile, Main tier. No constraint on a level is specified so as to allow this OMAF video profile to be used with HEVC Main 10 Profile at various levels.

All pictures in the bitstream shall be coded frames.

All the active SPSs of the bitstream shall be constrained as follows:

— general_progressive_source_flag shall be equal to 1.

— general_frame_only_constraint_flag shall be equal to 1.

— general_interlaced_source_flag shall be equal to 0.

When VUI is present, aspect_ratio_idc should not be present or aspect_ratio_idc should be equal to 0 (unspecified) or 1 (square).

NOTE 1 When aspect_ratio_idc is not present, Rec. ITU-T H.265 | ISO/IEC 23008-2 specifies that aspect_ratio_idc is inferred to be equal to 0.

NOTE 2 In order to merge tiles from different original bitstreams, init_qp_minus26 in all the PPSs of the original bitstreams is set to the same value in encoding.

10.1.7.3 ISO Base Media File Format constraints

10.1.7.3.1 Stream header

A Stream Header shall conform to subclause 8.6.1.2 and the following constraints:

— When a track is the only track in a file, `compatible_brands` containing a brand equal to `'siti'` in `FileTypeBox` indicates that the track conforms to this media profile. When a file contains multiple tracks, `compatible_brands` containing a brand equal to `'siti'` in `FileTypeBox` indicates that at least one of the tracks conforms to this media profile.

— A brand equal to `'siti'` in `FileTypeBox` indicates that the file contains one or more OMAF base tracks as specified in subclause 8.6 and as constrained by this subclause and that the OMAF base tracks reference OMAF tile tracks specified in subclause 8.6 and as constrained by this subclause.

— A track of this media profile shall be indicated to conform to this media profile through one or both of `FileTypeBox` and `TrackTypeBox`.

NOTE The ISOBMFF boxes and nesting, optionality and ordinality of the Stream Header are described in Table 53.

Table 53 — Stream Header for the simple tiling OMAF video profile

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	Format Req.	Specification	Constraints	Description
`ftyp`									1	ISO/IEC 14496-12:2020, subclause 4.3	This subclause	
Either `moov` or `!mov`									1	ISO/IEC 14496-12:2020, subclauses 8.2.1 and 8.19.6		
	`mvhd`								1	ISO/IEC 14496-12:2020, subclause 8.2.2	ISO/IEC 23000-19:2020, subclause 7.5.1	
	`mvex`								1	ISO/IEC 14496-12:2020, subclause 8.8.1		
		`mehd`							1	ISO/IEC 14496-12:2020, subclause 8.8.2	ISO/IEC 23000-19:2020, subclause 7.5.1	
		`trex`							1	ISO/IEC 14496-12:2020, subclause 8.8.3	ISO/IEC 23000-19:2020, subclause 7.5.14	

10.1.7.3.2 Track box format for OMAF base track and OMAF tile tracks

Each of the track boxes of the OMAF base track(s) and the OMAF tile tracks is constrained as follows:

— The `TrackBox` of each OMAF base track shall conform to subclause 8.6.1.3.

— The `TrackBox` of each OMAF tile track shall conform to subclause 8.6.1.4.

— At least one sample entry type of each sample entry of the track shall be equal to `'resv'`.

NOTE 1 `'resv'` does not have to be the track sample entry type, when the track has undergone several transformations. Consequently, this media profile can also be used when the track is protected.

— The untransformed sample entry type of each OMAF base track shall be equal to `'hvc2'`. Each OMAF base track shall conform to all constraints of the `'hvc2'` sample entry type, as specified in ISO/IEC 14496-15.

— The untransformed sample entry type of each OMAF tile track shall be equal to `'hvt1'`, `'hvt2'`, `'hvt3'` or `'hvc1'`. Each OMAF tile track shall conform to all constraints of the `'hvt1'`, `'hvt2'`, `'hvt3'` or `'hvc1'` sample entry type, as specified in ISO/IEC 14496-15.

— For each OMAF base track, the `scheme_type` values of `SchemeTypeBox` in the `RestrictedSchemeInfoBox` and of all instances of `CompatibleSchemeTypeBox` in the same `RestrictedSchemeInfoBox` shall include `'podv'` and at least one of `'ercm'` and `'erc2'`.

— For each OMAF tile track, either of the following shall be true:

  — The `scheme_type` values of `SchemeTypeBox` in the `RestrictedSchemeInfoBox` and of all instances of `CompatibleSchemeTypeBox` in the same `RestrictedSchemeInfoBox` include `'podv'` and at least one of `'erpv'` and `'ercm'`.

  - No `RestrictedSchemeInfoBox` is present in the sample entry and an `OverlayConfigBox` is present in the sample entry.

- Each OMAF base track shall contain either `'sabt'` or `'scal'` track references to OMAF tile tracks or to `'alte'` track groups containing OMAF tile tracks.

NOTE 2 When an OMAF base track contains `'scal'` track references to `'alte'` track groups, the tracks included in a referenced `'alte'` track group can represent different projected regions. The region-wise packing format of the OMAF base track is derived from the `RegionWisePackingBox`es of the OMAF tile tracks that an OMAF player selects from the referenced `'alte'` track groups. There can be several constructors within a sample of an extractor track that reference the same track reference index that points to the same `'alte'` track group. An OMAF player is expected to select OMAF tile tracks containing different projected regions for resolving the references from a sample of an extractor track to the same track reference index of an `'alte'` track group.

- When an OMAF base track references an OMAF tile track without `RegionWisePackingBox` and with `OverlayConfigBox` (either directly or indirectly through a reference to an `'alte'` track group), the OMAF base track shall be indicated to comply with the `'ovly'` toolset brand.

- When an OMAF base track contains `'sabt'` track references, the PPS(s) carried in the OMAF base track shall have tiles_enabled set to true, with num_tile_columns_minus1 and num_tile_rows_minus1 set to match the number of `'sabt'` track references associated with this OMAF base track.

- When an OMAF base track contains `'sabt'` track references, the projected regions of any two OMAF tile tracks referenced by the OMAF base track shall not overlap when both of the following conditions are true:

  - The OMAF tile tracks originate from the same projected picture without resampling.

  - The OMAF tile tracks are not in the same `'alte'` track group.

- `LHEVCConfigurationBox` shall not be present in `VisualSampleEntry`.

- `HEVCConfigurationBox` in `VisualSampleEntry` of each OMAF base track shall indicate conformance to the elementary stream constraints specified in subclause 10.1.6.2.

- A `SyncSampleBox` may be present for an OMAF base track.

The constraints specified in subclause 10.1.6.2 apply to each OMAF base track conforming to this media profile, where the bitstream based on which the constraints are derived is reconstructed from the OMAF base track as specified in subclause 10.1.7.3.3.

NOTE 3 The ISOBMFF boxes and nesting, optionality and ordinality of the `TrackBox` for the OMAF base track are described in Table 54.

Table 54 — Track box for OMAF base track for the simple tiling OMAF video profile

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	NL 9	Format Req.	Specification	Constraints	Description
	trak									1	ISO/IEC 14496-12:2020, subclause 8.3.1		
		tkhd								1	ISO/IEC 14496-12:2020, subclause 8.3.2	ISO/IEC 23000-19:2020, subclause 7.5.4 except that there are no constraints beyond those in ISO/IEC 14496-12 on the width and height fields and with the additional constraint that the field matrix shall always be set to the default values as defined in ISO/IEC 14496-12.	
		tref								1	ISO/IEC 14496-12:2020, subclause 8.3.3	This subclause	Track references to OMAF tile tracks.
		mdia								1	ISO/IEC 14496-12:2020, subclause 8.4		
			mdhd							1	ISO/IEC 14496-12:2020, subclause 8.4.2	ISO/IEC 23000-19:2020, subclause 7.5.5	
			hldr							1	ISO/IEC 14496-12:2020, subclause 8.4.3		
			minf							1	ISO/IEC 14496-12:2020, subclause 8.4.4		
				vmhd						1	ISO/IEC 14496-12:2020, subclause 12.1.2	ISO/IEC 23000-19:2020, subclause 7.5.6	
				dinf						1	ISO/IEC 14496-12:2020, subclause 8.7.1		
					dref					1	ISO/IEC 14496-12:2020, subclause 8.7.2		
						snim				1	ISO/IEC 14496-12:2020, subclause 8.7.2.2		

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	NL 9	Format Req.	Specification	Constraints	Description
				stbl						1	ISO/IEC 14496-12:2020, subclause 8.5	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stsd					1	ISO/IEC 14496-12:2020, subclause 8.5.2	ISO/IEC 23000-19:2020, subclause 7.5.10	
						resv				1			
						rinf				1	ISO/IEC 14496-12:2020, subclause 8.15.3	This subclause	
							hvc2			1	ISO/IEC 14496-15:2019, subclause 8.4	This subclause	
							schm			1	ISO/IEC 14496-12:2020, subclause 8.12.5		
							schi			1	ISO/IEC 14496-12:2020, subclause 8.12.6		
								povd		1	Subclause 7.6.2		
									prfr	1	Subclause 7.6.2		
									rwpk	0/1 for ercm. 1 for erc2.	Subclause 7.6.4	Subclause 7.6.1.4 for 'ercm' and subclause 7.6.1.7 for 'erc2'	
									rotn	0/1	Subclause 7.6.5		
									covi	0/1	Subclause 7.6.6		
								stvi		0/1	ISO/IEC 14496-12:2020, subclause 8.15.4.2	Subclause 7.6.1.2	
					stts					1	ISO/IEC 14496-12:2020, subclause 8.6.1.2	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stss					0/1	ISO/IEC 14496-12:2020, subclause 8.6.2		
					stsc					1	ISO/IEC 14496-12:2020, subclause 8.7.4	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stco					1	ISO/IEC 14496-12:2020, subclause 8.7.5	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stsz					1	ISO/IEC 14496-12:2020, subclause 8.7.3	ISO/IEC 23000-19:2020, subclause 7.5.12	

NOTE 3 The ISOBMFF boxes and nesting, optionality and ordinality of the `TrackBox` for each OMAF tile track referenced by the OMAF base track are described in Table 55.

Table 55 — Track box for OMAF tile track for the simple tiling OMAF video profile

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	NL 9	Format Req.	Specification	Constraints	Description
	`trak`									+	ISO/IEC 14496-12:2020, subclause 8.3.1		
		`tkhd`								1	ISO/IEC 14496-12:2020, subclause 8.3.2	ISO/IEC 23000-19:2020, subclause 7.5.4 except that there are no constraints beyond those in ISO/IEC 14496-12 on the `width` and `height` fields and with the additional constraint that the field `matrix` shall always be set to the default values as defined in ISO/IEC 14496-12.	
		`mdia`								1	ISO/IEC 14496-12:2020, subclause 8.4		
			`mdhd`							1	ISO/IEC 14496-12:2020, subclause 8.4.2	ISO/IEC 23000-19:2020, subclause 7.5.5	
			`hldr`							1	ISO/IEC 14496-12:2020, subclause 8.4.3		
			`minf`							1	ISO/IEC 14496-12:2020, subclause 8.4.4		
				`vmhd`						1	ISO/IEC 14496-12:2020, subclause 12.1.2	ISO/IEC 23000-19:2020, subclause 7.5.6	
				`dinf`						1	ISO/IEC 14496-12:2020, subclause 8.7.1		
					`dref`					1	ISO/IEC 14496-12:2020, subclause 8.7.2		
						`snim`				1	ISO/IEC 14496-12:2020, subclause 8.7.2.2		
				`stbl`						1	ISO/IEC 14496-12:2020, subclause 8.5	ISO/IEC 23000-19:2020, subclause 7.5.12	

NL 0	NL 1	NL 2	NL 3	NL 4	NL 5	NL 6	NL 7	NL 8	NL 9	Format Req.	Specification	Constraints	Description
					stsd					1	ISO/IEC 14496-12:2020, subclause 8.5.2	ISO/IEC 23000-19:2020, subclause 7.5.10	
						resv				1			
						rinf				1	ISO/IEC 14496-12:2020, subclause 8.15.3	This subclause	
							hvc1 hvt1 hvt2 hvt3			1	ISO/IEC 14496-15:2019, subclauses 8.4, 10.5, and 10.6	This subclause	
							schm			1	ISO/IEC 14496-12:2020, subclause 8.12.5		
							schi			1	ISO/IEC 14496-12:2020, subclause 8.12.6		
								povd		1	Subclause 7.6.2		
									prfr	1	Subclause 7.6.2		
									rwpk	0/1	Subclause 7.6.4	Subclause 7.6.1.3 for 'erpv' and subclause 7.6.1.4 for 'ercm'	
									rotn	0/1	Subclause 7.6.5		
									covi	0/1	Subclause 7.6.6		
								stvi		0/1	ISO/IEC 14496-12:2020, subclause 8.15.4.2	Subclause 7.6.1.2	
					stts					1	ISO/IEC 14496-12:2020, subclause 8.6.1.2	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stsc					1	ISO/IEC 14496-12:2020, subclause 8.7.4	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stss					0/1	ISO/IEC 14496-12:2020, subclause 8.6.2	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stco					1	ISO/IEC 14496-12:2020, subclause 8.7.5	ISO/IEC 23000-19:2020, subclause 7.5.12	
					stsz					1	ISO/IEC 14496-12:2020, subclause 8.7.3	ISO/IEC 23000-19:2020, subclause 7.5.12	

10.1.7.3.3 Other constraints

The following constraints apply when an OMAF base track is an extractor track and has an empty `RegionWisePackingBox`:

— When there is a sample in an OMAF base track whose extractors do not include the content of the referenced OMAF tile tracks in raster-scan order, the OMAF base track shall include a `SampleGroupDescriptionBox` of type `'trif'` and a `SampleToGroupBox` of type `'nalm'` with `grouping_type_parameter` equal to `'trif'`.

— When an OMAF base track includes a `SampleGroupDescriptionBox` of type `'trif'` and a `SampleToGroupBox` of type `'nalm'` with `grouping_type_parameter` equal to `'trif'`, the following applies:

  — At most one extractor shall be mapped to the same `groupID` in a `'nalm'` sample group description entry.

  NOTE An OMAF player ought to map the NAL units and NAL-unit-like structures to `groupID` values using a `'nalm'` sample group description entry mapped to a sample before resolving the extractors of the sample.

  — The following constraints apply in all `'trif'` sample group description entries of the OMAF base track:

    — `tile_region_flag` shall be equal to 1.

    — `independent_idc` shall be equal to 1 or 2.

    — `full_picture` shall be equal to 0.

    — `has_dependency_list` shall be equal to 0.

10.1.7.4 File decoding process

The inputs to the file decoding process are

— the `track_ID` value of an OMAF base track conforming to this media profile, and

— a file containing at least the OMAF base track with that `track_ID` value and all tracks that the OMAF base track directly or indirectly depends on.

An HEVC bitstream is generated from the track with the given `track_ID` value as specified in ISO/IEC 14496-15.

The HEVC bitstream shall conform to the elementary stream constraints specified in subclause 10.1.6.2.

The HEVC bitstream is decoded as specified in Rec. ITU-T H.265 (11/19) | ISO/IEC 23008-2:2020, subclause 8.1.1. The outputs of this process are the same as the outputs of Rec. ITU-T H.265 (11/19) | ISO/IEC 23008-2:2020, subclause 8.1.1. Additionally, for each decoded picture, this process outputs `ProjectionFormatBox`, `StereoVideoBox` (when applicable), the collection of the instances of `RegionWisePackingBox` of the tracks referenced by the OMAF base track and their spatial mapping to the decoded picture, and `RotationBox` (when applicable) that provide the input for the semantics of sample locations within the decoded picture as specified in subclause 7.5.1.

10.1.7.5 Expected OMAF player operation

OMAF players conforming to this media profile are expected to process all allowed boxes within the `SchemeInformationBox` for the `'erpv'`, `'ercm'` and `'erc2'` scheme types.

When receiving and playing a DASH presentation and a Representation containing initial viewing orientation metadata, as specified in subclause 7.7.4, is present and associated with a media Representation selected for playing, as specified in subclause 8.2.3, OMAF players are expected to receive and parse the initial viewing orientation metadata Representation and obey it when rendering the media Representation.

A player conforming to this OMAF video profile is expected to:

- Select an OMAF base track e.g. based on one or more of the following:
  - the picture size and picture rate of the OMAF base track;
  - the HEVC level indicated for the OMAF base track so that the decoding capacity is sufficient for decoding the bitstream resolved from the OMAF base track;
  - the absence/presence of the overlay toolset brand `'ovly'` in the `TrackTypeBox` of the OMAF base track or in the `profiles` parameter of the `@mimeType` attribute for the Adaptation Set carrying the OMAF base track, and the support of overlays in the OMAF player. OMAF players that do not support overlays should select an OMAF base track without the `'ovly'` toolset brand, when such an OMAF base track is available.
- When playing an OMAF file, conclude that an OMAF tile track contains background video when it contains a `RegionWisePackingBox`. For OMAF tile tracks containing background video, parse `SphereRegionQualityRankingBox` of the OMAF tile tracks referenced by the selected OMAF base track and select the OMAF tile tracks that cover the viewport at the highest available quality as indicated by in the instances of `SphereRegionQualityRankingBox`. The player is expected not to select such OMAF tile tracks that originate from the same projected picture without resampling and contain projected regions that overlap.
- When streaming OMAF content, conclude that a partial Adaptation Set contains background video when it contains a `RegionWisePackingBox` (in the Initialization Segment) or an RWPK, CC or SRQR descriptor. For Representations of a partial Adaptation Set containing background video, parse spherical region-wise quality ranking (SRQR) or content coverage (CC) descriptors, when present, and select the Adaptation Sets and Representations that cover the viewport at the highest available quality. For Representations of a partial Adaptation Set not containing background video, select a Representation from the partial Adaptation Set using a regular bitrate adaptation logic.
- Identify an OMAF tile track without `RegionWisePackingBox` and handle such an OMAF tile track in either of the following ways:
  - If an OMAF player supports the `'ovly'` toolset brand) and the OMAF tile track contains an `OverlayConfigBox`, the OMAF player is expected to handle the overlays carried in the OMAF tile track as specified by the overlay toolset brand.
  - Otherwise, the OMAF player is expected to ignore the decoded regions resulting from the decoding of the OMAF tile track.

10.2 Audio profiles

10.2.1 Overview

Subclause 10.2 defines media profiles for audio in OMAF. Table 56 provides an overview of the supported features. The detailed, specification for each audio profile is subsequently provided in the referenced subclause.

Table 56 — Overview of OMAF media profiles for audio

Media Profile	Codec	Profile	Level	Max Sampling Rate	3D Metadata	Brand	Subclause
OMAF 3D audio baseline profile	MPEG-H Audio	Low Complexity	1, 2 or 3	48 kHz	included in codec	`oabl`	10.2.2
OMAF 2D audio legacy profile	AAC	HE-AACv2	4	48 kHz	no 3D metadata	`oa2d`	10.2.3

NOTE The audio streams complying with the MPEG-H 3D Audio Low Complexity (LC) Profile, Levels 1 or 2 (i.e. `mpegh3daProfileLevelIndication` is set to "0x0B" or "0x0C", comply also with MPEG-H 3D Audio LC Profile, Level 3 (i.e. `mpegh3daProfileLevelIndication` is set to "0x0D"), as specified in ISO/IEC 23008-3:2019, subclause 5.3.2.

10.2.2 OMAF 3D audio baseline profile

10.2.2.1 General

This media profile fulfills the requirements to support 3D audio. Channels, objects and Higher-Order Ambisonics (HOA) are supported, as well as combinations of those. The profile is based on ISO/IEC 23008-3 (i.e. MPEG-H 3D Audio).

MPEG-H 3D Audio specifies coding of immersive audio material and the storage of the coded representation in an ISOBMFF track. The MPEG-H 3D Audio decoder has a constant latency, see ISO/IEC 23008-3:2019, Table 1. With this information, content authors can synchronize audio and video portions of a media presentation, e.g. ensuring lip-synch. When orientation sensor inputs (i.e. azimuth, elevation) of an MPEG-H 3D Audio decoder change, there will be some algorithmic and implementation latency (perhaps tens of ms) between user head movement and the desired sound field orientation. This latency will not impact audio/visual synchronization (i.e. lip synch), but only represents the lag of the rendered sound field with respect to the user head orientation.

MPEG-H 3D Audio specifies methods for binauralizing the presentation of immersive content for playback via headphones, as is needed for omnidirectional media presentations. MPEG-H 3D Audio specifies an interface for the user's viewing orientation and permits low-complexity, low-latency rendering of the audio scene to any user orientation.

10.2.2.2 Elementary stream constraints

The audio stream shall comply with the MPEG-H 3D Audio Low Complexity (LC) Profile, Levels 1, 2 or 3 as defined in ISO/IEC 23008-3:2019, subclause 4.8. The values of the `mpegh3daProfileLevelIndication` for LC Profile Levels 1, 2 and 3 are "0x0B", "0x0C" and "0x0D", respectively, as specified in ISO/IEC 23008-3:2019, subclause 5.3.2.

Audio data shall be encapsulated into MPEG-H Audio Stream (MHAS) packets according to ISO/IEC 23008-3:2019, Clause 14.

All MHAS packet types defined in ISO/IEC 23008-3:2019, Clause 14 may be present in the stream, except of the following packet types that shall not be present in the stream:

— `PACTYP_CRC16`

— `PACTYP_CRC32`

— `PACTYP_GLOBAL_CRC16`

— `PACTYP_GLOBAL_CRC32`

If Audio Scene Information per ISO/IEC 23008-3:2019, Clause 15 is present, it always shall be encapsulated in an MHAS PACTYP_AUDIOSCENEINFO packet. Audio Scene Information shall not be included in the `mpegh3daConfig()` structure in the MHAS PACTYP_MPEGH3DACFG packet.

10.2.2.3 ISO Base Media File Format constraints

10.2.2.3.1 General constraints

When a track is the only track in a file, `compatible_brands` containing a brand equal to `'oabl'` in `FileTypeBox` indicates that the track conforms to this media profile. When a file contains multiple tracks, `compatible_brands` containing a brand equal to `'oabl'` in `FileTypeBox` indicates that at least one of the tracks conforms to this media profile.

`compatible_brands` containing a brand equal to `'oabl'` in `TrackTypeBox` indicates that the track conforms to this media profile.

A track of this media profile shall be indicated to conform to this media profile through one or both of `FileTypeBox` and `TrackTypeBox`.

The sample entry `'mhm1'` shall be used for encapsulation of MHAS packets into ISOBMFF files, per ISO/IEC 23008-3:2019, subclause 20.6.

The sample entry `'mhm2'` shall be used in cases of multi-stream delivery, i.e. the MPEG-H Audio Scene is split into two or more streams for delivery as described in ISO/IEC 23008-3:2019, subclause 14.6.

If the `MHAConfigurationBox()` is present, the MPEG-H profile and level indicator `mpegh3daProfileLevelIndication` in the `MHADecoderConfigurationRecord()` shall be set to `"0x0B"`, `"0x0C"`, or `"0x0D"` for MPEG-H Audio LC Profile Level 1, Level 2, or Level 3, respectively, as specified in ISO/IEC 23008-3:2019, subclause 5.3.2.

The first sample of the movie and the first sample of every fragment (when applicable) shall be a Stream Access Point (SAP) of type 1 (i.e. sync sample). For MPEG-H Audio a sync sample shall be properly signalled according to ISO/IEC 14496-12. All rules defined in ISO/IEC 23008-3:2019, subclause 20.6.1 regarding sync samples shall apply. In, addition, a sync sample shall consist of MHAS packets in the following order:

— `PACTYP_MPEGH3DACFG`

— `PACTYP_AUDIOSCENEINFO` (if Audio Scene Information is present)

— `PACTYP_BUFFERINFO`

— `PACTYP_MPEGH3DAFRAME`

Additional MHAS packets may be present between the MHAS packets listed above or after the MHAS packet `PACTYP_MPEGH3DAFRAME`, with one exception: if present, the `PACTYP_AUDIOSCENEINFO` packet shall directly follow the `PACTYP_MPEGH3DACFG` packet, as defined in ISO/IEC 23008-3:2019, subclause 14.4.

MPEG-H Audio sync samples contain Immediate Playout Frames (IPFs), as specified in ISO/IEC 23008-3:2019, subclause 20.2, thus the audio data encapsulated in the MHAS packet `PACTYP_MPEGH3DAFRAME` shall contain the `AudioPreRoll()` syntax element, as defined in ISO/IEC 23008-3:2019, subclause 5.5.6, and shall follow the requirements for stream access points as defined in ISO/IEC 23008-3:2019, subclause 5.7. The audio configuration is delivered as part of the MHAS packet `PACTYP_MPEGH3DACFG` and, therefore, the `AudioPreRoll()` structure carried in the MHAS packet `PACTYP_MPEGH3DAFRAME` shall not contain the `Config()` structure, i.e. the `configLen` field of the `AudioPreRoll()` shall be equal to 0.

10.2.2.3.2 Configuration change constraints

A configuration change takes place in an audio stream when the content setup or the Audio Scene Information changes (e.g. when changes occur in the channel layout, the number of objects etc.), and therefore new `PACTYP_MPEGH3DACFG` and `PACTYP_AUDIOSCENEINFO` packets are required upon such occurrences. A configuration change usually happens at program boundaries, but it may also occur within a program.

The following constraints apply:

— At each configuration change, the `MHASPacketLabel` shall be changed to a different value from the `MHASPacketLabel` in use before the configuration change occurred. A configuration change may happen at the beginning of a new ISOBMFF file or at any position within the file. In the latter case, the File Format sample that contains a configuration change shall be encoded as a sync sample (RAP) as defined above.

— A sync sample that contains a configuration change and the last sample before such a sync sample may contain a truncation message (i.e. a `PACTYP_AUDIOTRUNCATION` packet in the MHAS stream) as defined in ISO/IEC 23008-3:2019, subclause 14.4. If MHAS packets of type `PACTYP_AUDIOTRUNCATION` are present, they shall be used as described in ISO/IEC 23008-3:2019, subclause 14.4.

ISOBMFF tracks that belong to one Audio Programme use different configurations and a switch between two ISOBMFF tracks represents also a configuration change. Thus, the `MHASPacketLabel` needs to have different values for all ISOBMFF tracks that belong to one Audio Programme. Also, after a configuration change the `MHASPacketLabel` needs to have different values for all ISOBMFF tracks comprising an Audio Programme.

10.2.2.3.3 Multi-stream constraints

The multi-stream-enabled MPEG-H Audio System is capable of handling Audio Programme Components delivered in several different elementary streams (e.g. the main MHAS stream containing one complete audio main, and one or more auxiliary MHAS streams, containing different languages and audio description). The MPEG-H Audio Metadata information (MAE) allows the MPEG-H Audio Decoder to correctly decode several MHAS streams.

The following constraints apply for file formats using the sample entry `'mhm2'`:

— One MHAS stream shall be the main stream, i.e. in exactly one MHAS stream the Audio Scene Information shall have the `mae_isMainStream` field set to 1. In all other MHAS streams the `mae_isMainStream` shall be set to 0.

— In each auxiliary MHAS stream (i.e. streams with `mae_isMainStream` field set to 0) the `mae_bsMetaDataElementIDoffset` field in the Audio Scene Information shall be set to the index of the first metadata element in the auxiliary MHAS stream minus one.

— All MHAS elementary streams that carry Audio Programme Components of one Audio Programme shall be time aligned.

— In each auxiliary MHAS elementary stream (i.e. streams with `mae_isMainStream` field set to 0), RAPs shall be aligned to the RAPs present in the main stream (i.e. the stream with `mae_isMainStream` field set to 1).

— Presentation Description Manifests need to make sure that all streams that contribute to one Audio Programme may be identified as such.

— For the main and the auxiliary MHAS stream(s), the `MHASPacketLabel` shall be set according to ISO/IEC 23008-3:2019, subclause 14.6. ISOBMFF tracks that belong to one Switching Set need to use different `MHASPacketLabel` values within the same range of values associated to one stream, as specified in ISO/IEC 23008-3:2019, subclause 14.6. For example, all ISOBMFF tracks in the Switching Set for the main stream use different values between 1 and 16, all ISOBMFF tracks in the Switching Set for the first auxiliary stream use values between 17 and 32, and so on.

10.2.2.3.4 Loudness and dynamic range control

Loudness metadata shall be embedded within the `mpegh3daLoudnessInfoSet()` structure as defined in ISO/IEC 23008-3:2019, subclause 6.3. Such loudness metadata shall include at least the loudness of the content rendered to the default rendering layout as indicated by the `referenceLayout` field (see ISO/IEC 23008-3:2019, subclause 5.3.2). More precisely, the `mpegh3daLoudnessInfoSet()` structure shall include at least one `loudnessInfo()` structure with `loudnessInfoType` set to 0, whose `drcSetId` and `downmixId` fields are set to 0 and which includes at least one `methodValue` field with `methodDefinition` set to 1 or 2 (see ISO/IEC 23008-3:2019, subclause 6.3.1, and ISO/IEC 23003-4:2020, subclause 7.3). The indicated loudness value shall be measured. It is presupposed that this is consistent with applicable legal requirements.

DRC metadata shall be embedded in the `mpegh3daUniDrcConfig()` and `uniDrcGain()` structures as defined in ISO/IEC 23008-3:2019, subclause 6.3. For each included DRC set the `drcSetTargetLoudnessPresent` field as defined in ISO/IEC 23003-4:2020, Clause 7 shall be set to 1.

The `bsDrcSetTargetLoudnessValueUpper` and `bsDrcSetTargetLoudnessValueLower` fields shall be configured to continuously cover the range of target loudness levels between -31 dB and 0 dB. The embedded DRC metadata should allow for a decoder output loudness of at least -16 LKFS.

Loudness compensation information (`mae_LoudnessCompensationData()`), as defined in ISO/IEC 23008-3:2019, subclause 15.5 shall be present in the Audio Scene Information if the `mae_allowGainInteractivity` field (according to ISO/IEC 23008-3:2019, subclause 15.3) is set to 1 for at least one group of audio elements.

10.2.3 OMAF 2D audio legacy profile

10.2.3.1 General

This media profile fulfills requirements to support 2D channel-based audio. The delivery of up to 5.1 channels is supported. The profile is based on MPEG-4 AAC specified in ISO/IEC 14496-3, which defines coding of general audio content. The delivery of up to 5.1 audio channels allows 2D rendering according to the user's viewing orientation.

HE-AAC is used worldwide in the most successful streaming services and supported by all major streaming and media platforms. Due to the wide reach, MPEG-4 AAC may be used for VR services and platforms, which use either mono, stereo, 4.0, or 5.1 surround channel configurations. The 2D Audio Legacy profile does not require any new signalling for the audio codec and its configuration. Therefore, it is compatible with all decoder implementations in the market.

10.2.3.2 Elementary stream constraints

10.2.3.2.1 General encoding constraints

The audio stream shall comply with MPEG-4 AAC-LC, HE-AAC or HE-AACv2 profiles, Level 4, as defined in ISO/IEC 14496-3.

For HE-AAC encoded tracks, the first sample of the ISOBMFF movie and the first sample of every ISOBMFF movie fragment (when applicable) shall be a SAP of type 1, notably, the SBR configuration information shall be present in the audio access unit.

ISOBMFF tracks containing AAC audio as defined in ISO/IEC 14496-3 shall conform to the following AAC audio encoding constraints:

— The elementary stream shall be a raw data stream, i.e. ADTS and ADIF headers shall not be present.

— Each AAC elementary stream shall be encoded using MPEG-4 AAC LC, HE-AAC, HE-AACv2, Level 4. Use of the MPEG-4 HE-AACv2 for stereo configuration is recommended for 32 kbps or lower.

- When using HE-AAC and HE-AACv2, explicit backwards compatible signalling shall be used to indicate the use of the SBR and PS coding tools.
- AAC elementary streams shall not exceed 48kHz sampling rate.
- The number of channels, including the LFE channel, of an AAC ISOBMFF track shall not exceed six audio channels.
- AAC ISOBMFF fragments containing HE-AAC shall start with a type 1 SAP, notably, the SBR configuration information shall be in the first packet.
- The transform length of the IMDCT for AAC shall be 1024 audio PCM samples for long blocks, and 128 audio PCM samples for short blocks.
- The following parameters shall not change within the elementary stream
    - Audio Object Type
    - Sampling Frequency
    - Channel Configuration

The `channelConfiguration` parameter carried in the `AudioSpecificConfig` shall be set according to one of the following specified values:

- `channelConfiguration` is set equal to 1 for mono audio.
- `channelConfiguration` is set equal to 2 for stereo audio.
- `channelConfiguration` is set equal to 4 for four channel audio.
- `channelConfiguration` is set equal to 5 for five channel audio.
- `channelConfiguration` is set equal to 6 for six channel audio, i.e. 5.1 audio.

Producing audio content capable of seamless bitrate adaptation with OMAF 2D Audio Legacy media profile (AAC-LC, HE-AAC, HE-AACv2) requires constrained encoding at fragment boundaries. For such scenarios, each AAC elementary stream shall be encoded following the constraints provided in ISO/IEC 23000-19:2020, subclause 10.5.2 to subclause 10.5.6.

Encoding recommendations for AAC audio tracks are provided in ISO/IEC 23000-19:2020, Annex G.

10.2.3.2.2 Syntax and values of syntactic elements

The syntax and values for syntactic elements shall conform to ISO/IEC 14496-3. The following element shall not be present in an MPEG-4 HE-AAC or HE-AACv2 elementary stream:

- coupling_channel_element (CCE)

If the program_config_element (PCE) element is present then it shall only list a set of channels corresponding to one of the fixed channel configurations specific in ISO/IEC 14496-3:2019, Table 1.19, and the element shall not change for the duration of the track.

The arrangement of syntactic elements shall be according to ISO/IEC 14496-3:2019, Table 1.19. For convenience, the arrangement of elements for the allowed channel configurations is reported in Table 57.

Table 57 — Arrangement of Audio syntactic elements

Channel Configuration	Number of Channels	Audio syntactic elements
1	1	<SCE>, <optional additional elements>, <TERM>, for HE-AAC v2, and mono HE-AAC or AAC-LC
2	2	<CPE>, <optional additional elements>, <TERM>, for stereo HE-AAC or AAC-LC
4	4	<SCE>, <CPE>, <SCE>, <optional additional elements>, <TERM>
5	5.0	<SCE>, <CPE>, <CPE>, <optional additional elements>, <TERM>
6	5.1	<SCE>, <CPE>, <CPE>, <LFE>, <optional additional elements>, <TERM>

NOTE Angled brackets (<>) are used above to indicate separate syntactic elements, not stream syntax.

The syntax and values for `individual_channel_stream` shall conform to ISO/IEC 14496-3. The following fields shall be set as follows:

— `gain_control_data_present` is set equal to 0.

10.2.3.2.3 AAC presentation timing

The AAC codec uses audio frames of a fixed length, and a transform which applies over two frames. To obtain correct audio from a frame, both frames in the transform are needed, and hence the prior encoded frame and the current encoded frame need to be decoded to output the first frame. This is sometimes called "priming" and may be signalled using the `'roll'` sample group.

A full reconstruction of the first encoded audio frame is sometimes not possible since there is no previous access unit. To still achieve a full reconstruction, a common practice is to add silence to the beginning of the audio signal. A more detailed explanation of this approach may be found in ISO/IEC 14496-24.

In practice, an encoder can prepend an arbitrary amount of (invalid) audio waveform samples to the signal. This portion of the audio signal is sometimes called "encoder delay" and varies depending on the implementation.

Presentation delay is compensated according to one of the following options:

— The most common approach to compensate for inserted extra audio is to add an offset edit list to the ISOBMFF header. In the case where padding has been added to the start of an audio stream, the `media_time` in the edit list is the length (in audio samples, as measured by the timescale of the track) of the inserted audio samples; 2112 is a common example for AAC.

— If the content has been generated according to ISO/IEC 23000-19:2020, Clause G.5, no `EditListBox` is present.

— If the SBR and PS coding tools are present, they shall not be considered for the purpose of delay compensation.

10.2.3.2.4 Loudness and dynamic range control

The audio stream should contain DRC and loudness metadata according to ISO/IEC 14496-3. The audio encoder should set the Program Reference Level to the loudness level of the audio stream.

The audio encoder should generate DRC metadata for light compression encoded in the `dyn_rng_ctl` and `dyn_rng_sgn` fields of `dynamic_range_info()` in the FIL element and DRC metadata for heavy compression in the `compression_value` field of `MPEG4_ancillary_data()` in the data stream element (DSE).

NOTE It is expected that the audio decoder will use the Program Reference Level, if available, to achieve a desired target loudness, if applicable. It is expected that the audio decoder will apply the DRC metadata, if present, according to ISO/IEC 14496-3 including the DRC Presentation Mode value of the `drc_presentation_mode` fields.

10.2.3.2.5 Maximum bitrate

The maximum bitrate of AAC elementary streams shall be calculated in accordance with the AAC buffer requirements as defined in ISO/IEC 14496-3:2019, subclause 4.5.3. Only the raw data stream shall be considered in determining the maximum bitrate (system-layer descriptors are excluded).

10.2.3.3 ISO Base Media File Format constraints

When a track is the only track in a file, `compatible_brands` containing a brand equal to `'oa2d'` in `FileTypeBox` indicates that the track conforms to this media profile. When a file contains multiple tracks, `compatible_brands` containing a brand equal to `'oa2d'` in `FileTypeBox` indicates that at least one of the tracks conforms to this media profile.

`compatible_brands` containing a brand equal to `'oa2d'` in `TrackTypeBox` indicates that the track conforms to this media profile.

A track of this media profile shall be indicated to conform to this media profile through one or both of `FileTypeBox` and `TrackTypeBox`.

The syntax and values of the `AudioSampleEntry` shall conform to `MP4AudioSampleEntry` (`'mp4a'`) as defined in ISO/IEC 14496-14. Table 58 lists the allowed AAC profiles.

Table 58 — AAC profiles

AAC profile	codingname	Sample entry
MPEG-4 AAC (AAC-LC)	mp4a	`MP4AudioSampleEntry`
MPEG-4 High Efficiency AAC (HE-AAC)	mp4a	`MP4AudioSampleEntry`
MPEG-4 High Efficiency AAC v2 (HE-AACv2)	mp4a	`MP4AudioSampleEntry`

The `SampleEntry` format in the `SampleDescriptionBox` is the same for each AAC audio profile.

10.2.3.3.1 Storage of AAC media samples

The following additional constraints apply:

— All audio media samples shall consist of one AAC audio access unit.

— All AAC access units in an ISOBMFF track shall be encoded with one of AAC LC, HE-AAC or HE-AACv2.

— The values given in `AudioSampleEntry`, `DecoderConfigDescriptor`, and `DecoderSpecificInfo` shall match the corresponding values in the AAC audio bitstream.

10.2.3.3.2 AAC audio sample entry

The syntax and values of the `AudioSampleEntry` shall conform to `MP4AudioSampleEntry` (`'mp4a'`) as defined in ISO/IEC 14496-14.

The sample entry and fields specified in this section shall not change within an ISOBMFF track.

The value of the `channelcount` parameter in the `AudioSampleEntry` box defined in ISO/IEC 14496-12 shall be set to one of the following specified values:

— `channelcount` is set equal to 1 for mono audio.

— `channelcount` is set equal to 2 for stereo audio.

— `channelcount` is set equal to 4 for four channel audio.

— `channelcount` is set equal to 5 for five channel audio.

— `channelcount` is set equal to 6 for six channel audio, i.e. 5.1 audio.

The value of the `channelcount` parameter in the `AudioSampleEntry` box shall correspond to the values of `channelConfiguration` field of `AudioSpecificConfig` according to Table 59:

Table 59 — Mapping of `channelcount` parameter in the `AudioSampleEntry` to `channelConfiguration` field of `AudioSpecificConfig`

`channelcount`	`channelConfiguration`
1	1
2	2
4	4
5	5
6	6
The channel to loudspeaker mapping for each `channelConfiguration` index is given in ISO/IEC 14496-3:2019, Table 1.19. The geometric speaker positions for `channelConfiguration = 4` (Quadrophonic speaker layout) is 0, 90°, -90°, 180° deg (azimuth)	

10.2.3.3.2.1 ES_Descriptor

The syntax and values for `ES_Descriptor` shall conform to ISO/IEC 14496-1, and the fields of the `ES_Descriptor` shall be set as follows:

— `ES_ID` is set equal to 0.

— `streamDependenceFlag` is set equal to 0.

— `URL_Flag` is set equal to 0.

— `OCRstreamFlag` is set equal to 0.

— `streamPriority` is set equal to 0.

— `decConfigDescr` is set equal to `DecoderConfigDescriptor`.

— `slConfigDescr` is set equal to `SLConfigDescriptor`, predefined type 2.

Descriptors other than those specified in subclauses 10.2.3.3.2.2 through 10.2.3.3.2.4 shall not be used.

10.2.3.3.2.2 DecoderConfigDescriptor

The syntax and values for `DecoderConfigDescriptor` shall conform to ISO/IEC 14496-1, and the fields of this descriptor shall be constrained to the following values.

— `decoderSpecificInfo` shall be used, and `ProfileLevelIndicationIndexDescriptor` shall not be used.

— `objectTypeIndication` is equal to 0x40 (Audio).

— `streamType` is equal to 0x05 (Audio Stream).

— `upStream` is equal to 0.

— `decSpecificInfo` is equal to `AudioSpecificConfig`.

10.2.3.3.2.3 AudioSpecificConfig

The syntax and values for `AudioSpecificConfig` shall conform to ISO/IEC 14496-3.

The following fields of `AudioSpecificConfig` shall be set according to ISO/IEC 14496-3 and subclause 10.2.3.2:

— `audioObjectType`

— `channelConfiguration`

— `extensionAudioObjectType`

— `GASpecificConfig`

10.2.3.3.2.4 GASpecificConfig

The syntax and values for `GASpecificConfig` shall conform to ISO/IEC 14496-3, and the fields of `GASpecificConfig` shall be set to the following values:

— `frameLengthFlag` is equal to 0 (1024 lines IMDCT).

— `dependsOnCoreCoder` is equal to 0.

— `extensionFlag` is equal to 0.

10.3 Image profiles

10.3.1 Overview

Subclause 10.3 defines OMAF media profiles for image coding. Table 60 provides an overview of the supported features. The detailed, specification for each image profile is provided in the referenced subclause. Common text for both profiles is provided in subclause 10.3.2.

Table 60 — Overview of OMAF media profiles for image coding

Media Profile	Codec	Profile	Level	Brand	Subclause
OMAF HEVC image profile	HEVC	Main 10	5.1	`heoi`	10.3.3
OMAF legacy image profile	JPEG	Not applicable	Not applicable	`jpoi`	10.3.4

10.3.2 Common specifications for image profiles

10.3.2.1 General

Subclause 10.3.2 specifies common constraints applicable to both OMAF image profiles. Both monoscopic and stereoscopic images up to 360 degrees are supported. These image profiles require the use of the equirectangular projection or the cubemap projection. When equirectangular projection and region-wise packing are in use, the number of regions in region-wise packing is required to be equal to 1 or 2 for monoscopic or stereoscopic content, respectively. When cubemap projection is in use, it is allowed to use up to one region per cube face in region-wise packing. The `'grid'` derived image item can be used to support large image widths and heights.

10.3.2.2 ISO Base Media File Format constraints for an image item conforming to an OMAF image profile

An image item is specified to conform to an OMAF image profile when all of the following constraints are true:

— Either one of the following applies:

  — The image item is a coded image item conforming to the OMAF image profile.

  — The image item is a `'grid'` derived image item, and each source image item of the derived image item is a coded image item conforming to the OMAF image profile.

— The item properties for the coded image item comply with the requirements and constraints specified in subclause 7.9.

— The item is not associated with any other types of essential item properties than those required by the item type of the image item, and `'stvi'`, `'prfr'`, `'rwpk'`, and `'rotn'`, which are specified in this document.

— The item is associated with a `'prfr'` item property that indicates the equirectangular or cubemap projection.

— When the item is associated with a `'rwpk'` item property, the values of the item property are constrained as follows:

  — When equirectangular projection is in use, the value of NumRegions, derived as specified in subclause 7.5.1.2, shall be equal to HorDiv1 * VerDiv1, derived as specified in subclause 7.9.3.1.

  — When cubemap projection is in use, the number of projected regions containing samples of a particular cube face shall not be greater than 1.

  NOTE When the image item covers less than 360 degrees, a cube face can be absent in a packed picture.

  — The value of PackingType[i] for each value of i in the range of 0 to NumRegions − 1, inclusive, derived as specified in subclause 7.5.1.2, shall be equal to 0.

10.3.2.3 ISO Base Media File Format constraints for a file conforming to an OMAF image profile

Each file including a four-character code of an OMAF image profile as a compatible brand shall conform to all of the following:

— The file shall include `'mif1'` among the compatible brands and comply with the requirements of `'mif1'` brand as specified in ISO/IEC 23008-12.

— The file shall contain at least one image item that conforms to all of the following:

  — The image item is present in the file.

  — When the image item is a derived image item, each source image item is present in the file.

  — The image item is either the primary item or any item from the alternate group containing the primary item.

  — The image item conforms to the OMAF image profile.

10.3.3 OMAF HEVC image profile

10.3.3.1 General

Subclause 10.3.3 specifies the OMAF HEVC image profile, which uses HEVC as the codec for coding of the image and follows the common constraints for OMAF image profiles.

10.3.3.2 Elementary stream constraints

The bitstream of a coded image item conforming to this media profile shall conform to HEVC Main 10 profile, Main tier, Level 5.1.

The bitstream contained in a coded image item shall consist of one and only one coded picture. The coded picture shall be a coded frame.

The active SPS of the bitstream shall be constrained as follows:

— general_progressive_source_flag shall be set to 1.

— general_frame_only_constraint_flag shall be set to 1.

— general_interlaced_source_flag shall be set to 0.

10.3.3.3 ISO Base Media File Format constraints

A coded image item is specified to conform to the `'heoi'` brand when all of the following constraints are true:

— The content of the coded image item conforms to the elementary stream constraints specified in subclause 10.3.3.2.

— The item has type `'hvc1'` and conforms to the requirements imposed by the `'hvc1'` item type, as specified in ISO/IEC 23008-12.

— The coded image item conforms to the constraints specified in subclause 10.3.2.2.

An image item is specified to conform to the `'heoi'` brand when it is a coded image item conforming to the `'heoi'` brand as specified above or a derived image item conforming to the constraints specified in subclause 10.3.2.2 and for which each source image item is a coded image item conforming to the `'heoi'` brand.

10.3.3.4 File decoding process

The inputs to the file decoding process are

— the `item_ID` value of the image item conforming to this media profile, and

— a file containing at least the image item.

If the image item with the given `item_id` value is a coded image item, the following applies:

— An HEVC bitstream consists of the content of the item with the given `item_ID` value.

— The HEVC bitstream shall conform to the elementary stream constraints specified in subclause 10.3.3.2.

— The HEVC bitstream is decoded as specified in Rec. ITU-T H.265 (11/19) | ISO/IEC 23008-2:2020, subclause 8.1.1.

— The outputs of this process are the same as the outputs of Rec. ITU-T H.265 (11/19) | ISO/IEC 23008-2:2020, subclause 8.1.1.

Otherwise (the image item is a `'grid'` derived image item), the following applies:

— For each source image item of the derived image item, the following applies:

  — An HEVC bitstream consists of the content of the source image item.

  — The HEVC bitstream shall conform to the elementary stream constraints specified in subclause 10.3.3.2.

  — The HEVC bitstream is decoded as specified in Rec. ITU-T H.265 (11/19) | ISO/IEC 23008-2:2020, subclause 8.1.1.

— The output of this process is formed by tiling the reconstructed images resulting from the decoding of all the source image items, as specified in the image grid derivation of ISO/IEC 23008-12.

Additionally, this process outputs `ProjectionFormatProperty`, `FramePackingProperty` (when applicable), `RegionWisePackingProperty` (when applicable), and `RotationProperty` (when applicable) that provide the input for the semantics of sample locations within the decoded picture as specified in subclause 7.5.1.

10.3.3.5 Recommendations and requirements for OMAF player

The requirements on readers conforming to the `'mif1'` brand, as specified in ISO/IEC 23008-12, shall be supported.

Players conforming to the `'heoi'` brand shall support displaying an image item that conforms to both of the following:

— The image item is either the primary item or any item from the alternate group containing the primary item.

— The image item conforms to the `'heoi'` brand as specified in subclause 10.3.3.3.

When displaying an image item conforming to the `'heoi'` brand, players are expected to obey semantics of the sample locations of the decoded picture as specified in subclause 7.5.1.

10.3.4 OMAF legacy image profile

10.3.4.1 General

Subclause 10.3.4 specifies OMAF legacy image profile, which uses JPEG as the codec for coding of the image and follows the common constraints for OMAF image profiles.

10.3.4.2 Elementary stream constraints

The elementary stream constraints are identical to those for the `'jpeg'` brand specified in ISO/IEC 23008-12.

10.3.4.3 ISO Base Media File Format constraints

A coded image item is specified to conform to the `'jpoi'` brand when all of the following constraints are true:

— The content of the coded image item conforms to the elementary stream constraints specified in subclause 10.3.4.2.

— The item has type `'jpeg'` and conforms to the requirements imposed by the `'jpeg'` item type, as specified in ISO/IEC 23008-12, or is coded with MIME type `'image/jpeg'` and conforms to that MIME type specification.

— The coded image item conforms to the constraints specified in subclause 10.3.2.2.

An image item is specified to conform to the `'jpoi'` brand when it is a coded image item conforming to the `'jpoi'` brand as specified above or a derived image item conforming to the constraints specified in 10.3.2.2 and for which each source image item is a coded image item conforming to the `'jpoi'` brand.

10.3.4.4 File decoding process

The inputs to the file decoding process are

— the `item_ID` value of the image item conforming to this media profile, and

— a file containing at least the image item.

If the image item with the given `item_id` value is a coded image item, the following applies:

— A JPEG bitstream consists of the content of the item with the given `item_ID` value.

— The JPEG bitstream shall conform to the elementary stream constraints specified in subclause 10.3.4.2.

— The JPEG bitstream is decoded as specified in ISO/IEC 10918-1.

— The output of this process is a decoded JPEG image.

Otherwise (the image item is a `'grid'` derived image item), the following applies:

— For each source image item of the derived image item, the following applies:

  — A JPEG bitstream consists of the content of the source image item.

  — The JPEG bitstream shall conform to the elementary stream constraints specified in subclause 10.3.4.2.

  — The JPEG bitstream is decoded as specified in ISO/IEC 10918-1.

— The output of this process is formed by tiling the reconstructed images resulting from the decoding of all the source image items, as specified in the image grid derivation of ISO/IEC 23008-12.

Additionally, this process outputs `ProjectionFormatProperty`, `FramePackingProperty` (when applicable), `RegionWisePackingProperty` (when applicable), and `RotationProperty` (when applicable) that provide the input for the semantics of sample locations within the decoded picture as specified in subclause 7.5.1.

10.3.4.5 Recommendations and requirements for OMAF player

The requirements on readers conforming to the `'mif1'` brand, as specified in ISO/IEC 23008-12, shall be supported.

Players conforming to the `'jpoi'` brand shall support displaying an image item that conforms to both of the following:

— The image item is either the primary item or any item from the alternate group containing the primary item.

— The image item conforms to the `'jpoi'` brand as specified in subclause 10.3.4.3.

When displaying an image item conforming to the `'jpoi'` brand, players are expected to obey semantics of the sample locations of the decoded picture as specified in subclause 7.5.1.

10.4 Timed text profiles

10.4.1 Overview

Subclause 10.4 defines media profiles for timed text. Timed text is used for providing subtitles and closed captions for omnidirectional video. Table 61 provides an overview of the supported features. The detailed, specification for each timed text profile is subsequently provided in the referenced subclause.

Table 61 — Overview of OMAF media profiles for timed text

Media Profile	Codec	Profile	Brand	Subclause
OMAF IMSC1 timed text profile	IMSC1	Text Profile or Image Profile	`ttml`	10.4.2
OMAF WebVTT timed text profile	WebVTT	n\a	`ttwv`	10.4.3

10.4.2 OMAF IMSC1 timed text profile

10.4.2.1 Elementary stream constraints

The elementary stream shall conform to the Text Profile or Image Profile specified in W3C Recommendation, *TTML Profiles for Internet Media Subtitles and Captions 1.0.1 (IMSC1)*.

10.4.2.2 ISO Base Media File Format constraints

When a track is the only track in a file, `compatible_brands` containing a brand equal to `'ttml'` in `FileTypeBox` indicates that the track conforms to this media profile. When a file contains multiple tracks, `compatible_brands` containing a brand equal to `'ttml'` in `FileTypeBox` indicates that at least one of the tracks conforms to this media profile.

compatible_brands containing a brand equal to 'ttml' in TrackTypeBox indicates that the track conforms to this media profile.

A track of this media profile shall be indicated to conform to this media profile through one or both of FileTypeBox and TrackTypeBox.

An IMSC1 track shall conforms to the IMSC1 track format as specified in subclause 7.10.3.

The media handler type is 'subt', and the track uses a subtitle media header.

The role of an IMSC1 track should be labelled by using the KindBox.

When timed text cues are displayed on sphere regions, the timed text sphere region metadata track, as specified in subclause 7.7.7, shall be present.

10.4.3 OMAF WebVTT timed text profile

10.4.3.1 Elementary stream constraints

The elementary stream shall conform to W3C Candidate Recommendation, *WebVTT, The Web Video Text Tracks Format*.

10.4.3.2 ISO Base Media File Format constraints

When a track is the only track in a file, compatible_brands containing a brand equal to 'ttwv' in FileTypeBox indicates that the track conforms to this media profile. When a file contains multiple tracks, compatible_brands containing a brand equal to 'ttwv' in FileTypeBox indicates that at least one of the tracks conforms to this media profile.

compatible_brands containing a brand equal to 'ttwv' in TrackTypeBox indicates that the track conforms to this media profile.

A track of this media profile shall be indicated to conform to this media profile through one or both of FileTypeBox and TrackTypeBox.

A WebVTT track shall conform to the WebVTT track format as specified in subclause 7.10.4, using a track handler_type of 'text' with a codingname of 'wvtt'.

The role of a WebVTT track should be labelled by using the KindBox.

When timed text cues are displayed on sphere regions, the timed text sphere region metadata track, as specified in subclause 7.7.7, shall be present.

11 Presentation profiles

11.1 OMAF viewport-independent baseline presentation profile

11.1.1 General

The OMAF viewport-independent baseline presentation profile is intended to provide the highest interoperability and quality on HMDs (including mobile-powered HMDs).

This profile fulfils the basic requirements to support 3D Audio and omnidirectional and 3D video. Both monoscopic and stereoscopic video are supported. The profile requires neither viewport-dependent delivery nor viewport-dependent decoding.

The profile also minimizes the options for basic interoperability.

11.1.2 ISO Base Media File Format constraints

An ISOBMFF file for which the content author considers that the VR experience is included in this file using the technologies for the OMAF viewport-independent baseline presentation profile may be offered using the ISOBMFF file brand `'ompp'`.

For a file with `compatible_brands` containing a brand equal to `'ompp'` in `FileTypeBox`, the following constraints apply:

— The file shall conform to the `'iso9'` brand.

— If containing video, the file shall contain at least one track conforming to the HEVC-based viewport-independent OMAF video profile as specified in subclause 10.1.2.

— If containing audio, the file shall contain at least one track conforming to the OMAF 3D audio baseline profile as specified in subclause 10.2.2.

11.2 OMAF viewport-dependent baseline presentation profile

11.2.1 General

The OMAF viewport-dependent baseline presentation profile is intended to provide interoperability and quality on the HMDs that go beyond the viewport resolution achievable by the OMAF viewport-independent baseline presentation profile.

This profile fulfils requirements to support 3D audio and omnidirectional and 3D video. Both monoscopic and stereoscopic video are supported. The profile requires viewport-dependent delivery and rendering.

11.2.2 ISO Base Media File Format constraints

An ISOBMFF file containing a VR experience using the technologies for the OMAF viewport-dependent baseline presentation profile may be offered using the ISOBMFF file brand `'ovdp'`.

For a file with `compatible_brands` containing a brand equal to `'ovdp'` in `FileTypeBox`, the following constraints apply:

— The file shall conform to the `'iso9'` brand.

— If containing video, the file shall contain at least one track conforming to the HEVC-based viewport-dependent OMAF video profile as specified in subclause 10.1.3.

— If containing audio, the file shall contain at least one track conforming to the OMAF 3D audio baseline profile as specified in subclause 10.2.2.

12 OMAF toolset brands

12.1 Overlay toolset brand

12.1.1 Overview

This toolset brand specifies overlay-related requirements on files and OMAF players.

12.1.2 ISO Base Media File Format constraints

The following constraints shall be obeyed in files containing the brand `'ovly'` among `compatible_brands` of the `FileTypeBox`:

— The file shall contain one or more `'ovbg'` entity groups.

— When the `OverlayPriority` control structure is present for an overlay and `overlay_priority` in `OverlayPriority` is equal to 0, `overlay_control_essential_flag[i]` shall be absent or equal to 0 for all values of `i` greater than 13 in the `SingleOverlayStruct()` of the overlay.

12.1.3 OMAF player operation

When an OMAF player conforming to this toolset brand is given a file containing the brand `'ovly'` among `compatible_brands` of the `FileTypeBox` as input, the OMAF player is expected to render the background visual media and the overlays of an `'ovbg'` entity group as specified for the expected behaviour in Annex G.

12.2 Viewpoint toolset brand

12.2.1 Overview

This toolset brand specifies viewpoint-related requirements on files and OMAF players.

12.2.2 ISO Base Media File Format constraints

The following constraints shall be obeyed in files containing the brand `'vwpt'` among `compatible_brands` of the `FileTypeBox`:

— The file shall contain two or more `'vipo'` entity groups.

12.2.3 OMAF player operation

When an OMAF player conforming to this toolset brand is given a file containing the brand `'vwpt'` among `compatible_brands` of the `FileTypeBox` as input, the OMAF player is expected to handle viewpoints as specified for the expected behaviour in Annex G.

12.3 Non-linear storyline toolset brand

12.3.1 Overview

This toolset brand specifies requirements for authoring files suitable for non-linear storyline and the corresponding requirements for OMAF players.

12.3.2 ISO Base Media File Format constraints

The following constraints shall be obeyed in files containing the brand `'nlsl'` among `compatible_brands` of the `FileTypeBox`:

— The file shall contain two or more `'vipo'` entity groups.

— If a first `'vipo'` entity group does not contain the `ViewpointSwitchingListStruct()`, there shall be a second `'vipo'` entity group for which at least one of the following constraints is met:

  — the `ViewpointSwitchingListStruct()` is present and at least one `destination_viewpoint_id` value is equal to the identifier of the first `'vipo'` entity group.

  — the `ViewpointLoopingStruct()` is present and the `loop_dest_viewpoint_id` value is equal to the identifier of the first `'vipo'` entity group.

— For each viewpoint switch signalled in the `ViewpointSwitchingListStruct()`, the flag `viewpoint_switch_region_flag` shall be equal to 1.

12.3.3 OMAF player operation

When an OMAF player conforming to this toolset brand is given a file containing the brand `'nlsl'` among `compatible_brands` of the `FileTypeBox` as input, the OMAF player is expected to handle viewpoints as specified for the expected behaviour in Clause G.8.

Annex A

(normative)

OMAF DASH schema

The OMAF DASH schema for descriptors defined in this document is provided below.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:mpeg:mpegI:omaf:2017"
  xmlns:omaf="urn:mpeg:mpegI:omaf:2017"
  elementFormDefault="qualified">
  <xs:element name="sphRegionQuality" type="omaf:SphRegionQualityType"/>
  <xs:complexType name="SphRegionQualityType">
    <xs:sequence>
      <xs:element name="qualityInfo" type="omaf:QualityInfoType" minOccurs="1" maxOccurs="255"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="shape_type" type="xs:unsignedByte" use="optional" default="0"/>
    <xs:attribute name="remaining_area_flag" type="xs:boolean" use="optional" default="0"/>
    <xs:attribute name="view_idc_presence_flag" type="xs:boolean" use="optional" default="0"/>
    <xs:attribute name="quality_ranking_local_flag" type="xs:boolean" use="optional" default="0"/>
    <xs:attribute name="quality_type" type="omaf:QualityType" use="required"/>
    <xs:attribute name="default_view_idc" type="omaf:ViewType" use="optional"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <xs:complexType name="QualityInfoType">
    <xs:attribute name="quality_ranking" type="xs:unsignedByte" use="required"/>
    <xs:attribute name="view_idc" type="omaf:ViewType" use="optional"/>
    <xs:attribute name="orig_width" type="xs:unsignedShort" use="optional"/>
    <xs:attribute name="orig_height" type="xs:unsignedShort" use="optional"/>
    <xs:attribute name="centre_azimuth" type="omaf:Range1" use="optional"/>
    <xs:attribute name="centre_elevation" type="omaf:Range2" use="optional"/>
    <xs:attribute name="centre_tilt" type="omaf:Range1" use="optional"/>
    <xs:attribute name="azimuth_range" type="omaf:HRange" use="optional"/>
    <xs:attribute name="elevation_range" type="omaf:VRange" use="optional"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <xs:element name="twoDRegionQuality" type="omaf:twoDRegionQualityType"/>
  <xs:complexType name="twoDRegionQualityType">
    <xs:sequence>
      <xs:element name="twoDqualityInfo" type="omaf:twoDQualityInfoType" minOccurs="1"
maxOccurs="255"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="remaining_area_flag" type="xs:boolean" use="optional" default="0"/>
    <xs:attribute name="view_idc_presence_flag" type="xs:boolean" use="optional" default="0"/>
    <xs:attribute name="quality_ranking_local_flag" type="xs:boolean" use="optional"   default="0"/>
  <xs:attribute name="quality_type" type="omaf:QualityType" use="required"/>
  <xs:attribute name="default_view_idc" type="omaf:ViewType" use="optional"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<xs:complexType name="twoDQualityInfoType">
  <xs:attribute name="quality_ranking" type="xs:unsignedByte" use="required"/>
  <xs:attribute name="view_idc" type="omaf:ViewType" use="optional"/>
  <xs:attribute name="orig_width" type="xs:unsignedShort" use="optional"/>
  <xs:attribute name="orig_height" type="xs:unsignedShort" use="optional"/>
  <xs:attribute name="left_offset" type="xs:unsignedShort" use="optional"/>
  <xs:attribute name="top_offset" type="xs:unsignedShort" use="optional"/>
  <xs:attribute name="region_width" type="xs:unsignedShort" use="optional"/>
  <xs:attribute name="region_height" type="xs:unsignedShort" use="optional"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<xs:element name="cc" type="omaf:CCType"/>
<xs:complexType name="CCType">
  <xs:sequence>
    <xs:element name="coverageInfo" type="omaf:coverageInfoType" minOccurs="1" maxOccurs="255"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="shape_type" type="xs:unsignedByte" use="optional" default="0"/>
  <xs:attribute name="view_idc_presence_flag" type="xs:boolean" use="optional" default="0"/>
  <xs:attribute name="default_view_idc" type="omaf:ViewType" use="optional"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
<xs:complexType name="coverageInfoType">
  <xs:attribute name="view_idc" type="omaf:ViewType" use="optional"/>
  <xs:attribute name="centre_azimuth" type="omaf:Range1" use="optional" default="0"/>
  <xs:attribute name="centre_elevation" type="omaf:Range2" use="optional" default="0"/>
  <xs:attribute name="centre_tilt" type="omaf:Range1" use="optional" default="0"/>
  <xs:attribute name="azimuth_range" type="omaf:HRange" use="optional" default="23592960"/>
  <xs:attribute name="elevation_range" type="omaf:VRange" use="optional" default="11796480"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<xs:attribute name="projection_type" type="omaf:listOfUnsignedByte"/>
<xs:simpleType name="listOfUnsignedByte">
  <xs:restriction>
    <xs:simpleType>
      <xs:list itemType="xs:unsignedByte"/>
    </xs:simpleType>
    <xs:minLength value="1"/>
  </xs:restriction>
</xs:simpleType>

<xs:attribute name="packing_type" type="omaf:OptionallistOfUnsignedByte"/>
<xs:simpleType name="OptionallistOfUnsignedByte">
  <xs:restriction>
```

```
        <xs:simpleType>
          <xs:list itemType="xs:unsignedByte"/>
        </xs:simpleType>
        <xs:minLength value="0"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:attribute name="view_dimension_idc" type="omaf:viewDIdcType"/>
    <xs:simpleType name="viewDIdcType">
      <xs:restriction base="xs:unsignedByte">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="7"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="Range1">
      <xs:restriction base="xs:int">
        <xs:minInclusive value="-11796480"/>
        <xs:maxInclusive value="11796479"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="Range2">
      <xs:restriction base="xs:int">
        <xs:minInclusive value="-5898240"/>
        <xs:maxInclusive value="5898240"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="HRange">
      <xs:restriction base="xs:unsignedInt">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="23592960"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="VRange">
      <xs:restriction base="xs:unsignedInt">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="11796480"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="QualityType">
      <xs:restriction base="xs:unsignedByte">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="15"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="ViewType">
      <xs:restriction base="xs:unsignedByte">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="3"/>
      </xs:restriction>
```

```
  </xs:simpleType>
</xs:schema>
```