**ISO/IEC JTC 1/SC 29/WG 03 N0577**

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

**Document type:** Output Document

**Title:** Technologies under Consideration for OMAF

**Status:** Approved

**Date of document:** 2022-05-09

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

**No. of pages:** 46 (with cover page)

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

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

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

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

**ISO/IEC JTC 1/SC 29/WG 03 N0577**

**January 2022, Virtual**

|  |  |
| --- | --- |
| **Title** | **Technologies under Consideration for OMAF** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **21470** |

# Introduction

This document includes technologies under consideration for ISO/ IEC 23090-2 (OMAF)

This includes the following:

1. Additional Signalling for viewing space in OMAF (from m43960)
2. Compact region-wise packing signaling (from m43436)
3. Extending the 'cdtg' track reference type to support referring to both tracks and items (from m43422)
4. Support of transparent background in OMAF (option 1 of m47223)
5. OMAF late binding DASH streaming application model
6. Recommended viewport signaling (m50944)
7. Richer Interaction Signalling for overlays (m50883)
8. Signalling for a track/image item containing both background and overlay (m52238)
9. Viewpoint switching behavior flag (m53599)
10. Indicating the OMAF tile track layout within the decoded pictures resulting from an OMAF base track (m54413 aspect 2)
11. Tile-based viewport-dependent streaming (m59581)

# Additional Signaling for Viewing space

This clause includes the technology under consideration for viewing space including:

* Signaling of excluded regions

for a given viewing space.

The blue highlighted text includes the additional signaling technology for viewing space. Text in black font was agreed to be included in the Working Draft of OMAF 2nd Ed. in MPEG#123 meeting. Text regarding guard ranges was agreed to be included in the DIS of OMAF 2nd Ed. in MPEG#129 meeting.

## Guard range and excluded regions

The content creator may include guard range and excluded region information in order to have a fine-grained viewing space. Figure 1 illustrates the guard ranges on the cuboid viewing space (VS) information:

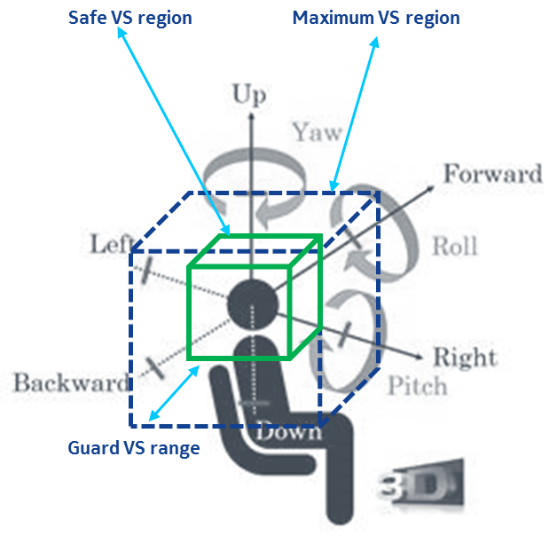


Figure 1: Partitions of a cuboid viewing space (VS)

* **Safe VS region** in X, Y, Z direction (green box): The viewing space or region that can be safely assumed to be responsive to the viewer’s translational movement. Responsiveness may mean rendering an optimized synthesized or selected view which represents the spatial features of the content from the viewing position (e.g. occlusion and parallax handling)
* **Maximum VS region** in X, Y, Z direction (dashed blue box): maximum allowed region where the content is responsive to head motion and immersive VR experience is supported (except for the excluded regions described below). The best visual experience may not be guaranteed outside the *safe VS region* due to limitations of the content.
* **Guard VS range:** The range between safe and maximum VS regions. This region is not guaranteed to provide the best viewing experience. A visually non-abrupt and graceful degradation could be applied by the renderer when the view point is in the *guard VS range*.
* **Excluded regions:** The viewing space may include excluded spaces indicated by boxes where the immersive VR experience is not supported.

[Editor’s Note (EA):

OMAF requirements traceability to guard range and excluded regions:

Guard range and excluded regions are designed to support the following MPEG-I Phase 1b requirements:

* 1. The Specification shall enable content authoring in a manner that Head-Motion Parallax and Binocular Disparity are supported in rendering without annoying artefacts while the user is in the Viewing Space.

Note: Testing methodology for visual quality of experience of rendering with Head-Motion Parallax and Binocular Disparity should be developed and/or chosen during the standardization.

* + 1. The Specification shall enable the support of Head-Motion Parallax and Binocular Disparity in rendering in a manner that any potential degradations in the quality of experience are not abrupt but graceful.
    2. The Specification shall enable an OMAF player implementation to detect viewing positions and/or viewing orientations where rendering is potentially unacceptable.

Note: Unacceptable rendering may occur when the viewing position is outside a Viewing Space and/or when the viewing orientation is outside the range of valid viewing orientations as discussed in 1.1.1

Note: OMAF player implementations could indicate to the users when the viewing position/orientation is such that it is close to yield an unacceptable rendering quality. For example, the OMAF player may fade the image to grey if the head moves too far away from the sweet spot.

The above-listed OMAF v2 requirements clearly indicate a need for defining regions where:

-a graceful visual quality degradation is possible. Such regions can be efficiently signaled via guard ranges.

-visual quality is unacceptable and hence no rendering is allowed. Excluded regions can efficiently signal such regions]

## ISOBMFF Viewing Space Box:

### ViewingSpaceStruct

aligned(8) class ViewingSpaceStruct(i) {  
 unsigned int(8) viewing\_space\_shape\_type;   
 unsigned int(16) distance\_scale;  
 bit(1) guard\_range\_indicator;  
 bit(6) reserved;   
 if(viewing\_space\_shape\_type==0){  
 VRBB(i)  
 if (guard\_range\_indicator) {  
 unsigned int(8) guard\_range\_X;  
 unsigned int(8) guard\_range\_Y;  
 unsigned int(8) guard\_range\_Z;  
 }  
 }  
 else if(viewing\_space\_shape\_type==1)   
 SphereStruct();  
 else if(viewing\_space\_shape\_type==2)   
 CylinderStruct();   
 else if(viewing\_space\_shape\_type==3)   
 EllipsoidStruct();  
}

aligned(8) class VRBB(i) {  
 signed int(32) xMin[i];   
 signed int(32) xMax[i];  
 signed int(32) yMin[i];   
 signed int(32) yMax[i];   
 signed int(32) zMin[i];  
 signed int(32) zMax[i];   
 }   
aligned(8) SphereStruct() {  
 unsigned int (32) radius;   
 if(guard\_range\_indicator){   
 bit(1) reserved;  
 unsigned int (7) guard\_radius\_diff;   
 }  
 }

aligned(8) class CylinderStruct() {  
 unsigned int (32) cylinder\_radius;   
 Point(0);   
 Point(1);   
 if(guard\_range\_indicator){   
 unsigned int (8) cylinder\_guard\_radius\_diff;   
 }  
}

aligned(8) class Point(i) {  
 signed int(32) x[i];   
 signed int(32) y[i];  
 signed int(32) z[i];   
}

aligned(8) class EllipsoidStruct() {  
 unsigned int (32) lengthX;  
 unsigned int (32) lengthY;  
 unsigned int (32) lengthZ;

if(guard\_range\_indicator){   
 unsigned int (8) guard\_lenghthX\_diff;

unsigned int (8) guard\_lenghthY\_diff;

unsigned int (8) guard\_lenghthZ\_diff;  
 }  
}

#### 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.

distance\_scaleis a positive integer value which indicates the units corresponding to 1 cm.

guard\_range\_indicatoris a boolean value which indicates whether or not a guard range information is present in the ViewingSpaceStruct.

guard\_range\_X, guard\_range\_Y and guard\_range\_Zindicate the guard ranges in percentage of the maximum viewing space ranges per axis indicated by Abs(xMax-xMin), Abs(yMax-yMin) and Abs(zMax-zMin) respectively. Guard ranges apply to both minimum and maximum values per axis as indicated by VRBB(0). A value of 0 indicates that there is no guard range present for a particular axis. Values greater than 50 are not allowed and reserved. The absolute value Abs(x) operation is defined as follows:

Abs( x )  

xMin, yMin and zMin is a 16.16 fixed-point value in distance scale that specifies the minimum value of X, Y, Z co-ordinates respective to the center point of the viewing space.

xMax, yMax and zMax is a 16.16 fixed-point value in distance scale that specifies the maximum value of X, Y, Z co-ordinates respective to the center point of the viewing space.

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 space spherical shell inside the sphere of radius equal to radius as a percentage of the radius. guard\_radius\_diff shall be less than radius. guard\_radius\_diff equal to 0 indicates that the guard range space is not present and immersive VR experience is guaranteed in the sphere of radius equal to radius. guard\_radius\_diff shall be in the range of 0 to 100, inclusive. Value 101 to 127 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 space 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 space is not present and immersive VR experience is guaranteed in the cylinder defined by Point(0), Point(1), and cylinder\_radius. cylinder\_guard\_radius\_diff shall be in the range of 0 to 100. Value 101 to 255 are reserved.

x[i], y[i] and z[i] is a value in units of distance scale that specifies the X, Y, Z co-ordinates of a Point in 3D space with respect to the center point of the viewing space.

lengthX, lengthY, and lengthZ specify respectively the semi-axes lengths of X, Y and Z axis of an ellipsoid which has the same center as the viewing space, in units of 2−16 millimeters. lengthX, lengthY, and lengthZ shall be in the ranges of 1 to 65 536 \* 216 – 1 (i.e., 4 294 967 295), inclusive.

guard\_lengthX\_diff, guard\_lengthY\_diff, guard\_lengthZ\_diff specify the thickness of the guard range space between outer ellipsoid with semi-axes lengths in X, Y and Z equal to lengthX, lengthY, lengthZ respectively and inner ellipsoid with semi-axes lengths in X, Y and Z as a guard\_lengthX\_diff, guard\_lengthY\_diff, guard\_lengthZ\_diff percentage of the specified lengthX, lengthY, lengthZ values respectively. The values shall be in the range of 0 to 100. Value 101 to 255 are reserved

When viewpoint position information is present, the center position of the viewing space is equal to the position of the viewpoint. When viewpoint position information is not present, the center position of the viewing space is equal to (0,0,0) in the common reference coordinate system. In both cases, the X,Y,Z coordinate axis are aligned with the reference global coordinate axis.

### Viewing Space box

#### Definition

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

The fields in this box specify 3D viewing space within which an immersive VR experience is provided.

#### Syntax

aligned(8) class ViewingSpaceBox extends FullBox('vssn', 0, flags) {  
 unsigned int(1) vr\_space\_exclusions\_info\_present\_flag; ;   
 bit(7) reserved = 0;   
 ViewingSpaceStruct(0);   
 if (vr\_space\_exclusions\_info\_present\_flag == 1) {  
 unsigned int(8) num\_excluded\_regions\_minus1;   
 for (i = 1; i <= num\_excluded\_regions\_minus1+1; i++)   
 VRBB(i)   
 }  
}

#### Semantics

VRBB(0) specifies the bounding box (X, Y, Z) minimum and maximum co-ordinates which specify the 3D viewing space within which excluding any signaled excluded VR space areas an immersive VR experience is supported.

vr\_space\_exclusions\_info\_present\_flag equal to 1 specifies that the immersive VR experience space includes excluded areas where immersive VR experience is not supported within the bounding box VRBB(0). vr\_space\_exclusions\_info\_present\_flag equal to 0 specifies that the immersive VR experience space does not include any excluded areas within the bounding box VRBB(0).

num\_excluded\_regions\_minus1 plus 1 specifies the number of VRBB(i) structures signaled which indicate the number of excluded spaces from the immersive VR experience.

VRBB(i) for i greater than 0 specifies the bounding box (X, Y, Z) minimum and maximum co-ordinates which specify the 3D space in the 3D space specified by VRBB(0) within which an immersive VR experience is not supported. It is a requirement of conformance that each of VRBB(i) for i greater than 0 shall be within the 3D space specified by VRBB(0).

## Time varying viewing space signaling

The Immersive VR viewing space timed metadata track indicates the VR viewing space boundaries where immersive experience is supported. The space can be static or can change dynamically on sample basis. A bounding box indicates the limits of the immersive VR space. Certain spaces within the bounding box could be indicated as excluded spaces for VR experience. For example the space within the environment which has capture time limitations and immersive VR experience by the camera/ microphone capture equipment may be marked as excluded space within the overall immersive VR space. In other example the excluded space signaling can be used if the immersive VR experience is provided in non rectangular prism/ cuboid. There may be also other content generation/ application / tracking equipment specific reasons for excluded space.

### Time varying Immersive VR viewing space Signaling

#### Definition

The Immersive VR viewing space timed metadata track indicates the VR viewing space boundaries where immersive experience is supported. The space can be static or can change dynamically on sample basis. A bounding box indicates the limits of the immersive VR space. Certain spaces within the bounding box could be indicated as excluded spaces for VR experience.

#### Syntax and semantics

The track sample entry type ‘vrsp’ shall be used. The sample entry is specified as follows:

class VRSpaceSampleEntry(type) extends MetadataSampleEntry(‘vrsp’) {  
 unsigned int(1) static\_vr\_space\_flag;   
 unsigned int(1) vr\_space\_exclusions\_info\_present\_flag;   
 if (vr\_space\_exclusions\_info\_present\_flag == 1) {  
 unsigned int(1) static\_vr\_exclusions\_flag;   
 bit(5) reserved = 0;   
 }  
 else {  
 bit(6) reserved = 0;  
 }  
 if (static\_vr\_space\_flag == 1) {  
 ViewingSpaceStruct(0);   
 }   
 if (vr\_space\_exclusions\_info\_present\_flag == 1) {  
 if (static\_vr\_exclusions\_flag == 1) {  
 unsigned int(8) static\_num\_excluded\_regions\_minus1;   
 for (i = 1; i <= static\_num\_excluded\_regions\_minus1+1; i++)   
 VRBB(i)   
 }  
 }  
}

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.

vr\_space\_exclusions\_info\_present\_flag equal to 1 specifies that the immersive VR experience viewing space includes excluded areas where immersive VR experience is not supported within the bounding box VRBB(0) for immersive VR experience. vr\_space\_exclusions\_info\_present\_flag equal to 0 specifies that the immersive VR experience viewing space does not include any excluded areas within the bounding box VRBB(0) for immersive VR experience.

static\_vr\_exclusions\_flag equal to 1 specifies that the space excluded from immersive VR experience does not change for each sample referring to this sample entry. static\_vr\_exclusions\_flag equal to 0 specifies that the space excluded from immersive VR experience may change for samples referring to this sample entry. When not present static\_vr\_exclusions\_flag is inferred to be equal to 1.

VRBB(0) in the sample entry specifies the bounding box (X, Y, Z) minimum and maximum co-ordinates which specify the 3D space within which excluding any signaled excluded VR space areas an immersive VR experience is supported.

static\_num\_excluded\_regions\_minus1 plus 1 specifies the number of VRBB(i) structures signaled in the sample entry which indicate the number of excluded spaces from the immersive VR experience.

VRBB(i) for i greater than 0 in the sample entry specifies the bounding box (X, Y, Z) minimum and maximum co-ordinates which specify the 3D space in the 3D space specified by VRBB(0)within which an immersive VR experience is not supported. It is a requirement of conformance that each of VRBB(i) for i greater than 0 shall be within the 3D space specified by VRBB(0).

The sample syntax shown in VRSpaceSample shall be used.

aligned(8) VRSpaceSample() {  
 if(static\_vr\_space\_flag == 0)   
 ViewingSpaceStruct(0);  
 if (vr\_space\_exclusions\_info\_present\_flag == 1) {  
 if (static\_vr\_exclusions\_flag == 0) {  
 unsigned int(8) num\_excluded\_regions\_minus1;   
 for (i = 1; i <= num\_excluded\_regions\_minus1+1; i++)   
 VRBB(i)   
 }  
 }  
}

VRBB(0) in the sample specifies the bounding box (X, Y, Z) minimum and maximum co-ordinates which specify the 3D space within which excluding any signaled excluded VR space areas an immersive VR experience is supported for the associated media track samples.

num\_excluded\_regions\_minus1 plus 1 specifies the number of VRBB(i) structures signaled in the sample which indicate the number of excluded spaces from the immersive VR experience.

VRBB(i) for i greater than 0 in the sample specifies the bounding box (X, Y, Z) minimum and maximum co-ordinates which specify the 3D space in the 3D space specified by VRBB(0)within which an immersive VR experience is not supported for the associated media track samples. It is a requirement of conformance that each of VRBB(i) for i greater than 0 shall be within the 3D space specified by VRBB(0).

# Compact Signaling of Region-Wise Packing

The highlighted text includes the additional signaling for compact region-wise packing.

[Ed. Note: It is noted that the syntax below may not be backward compatible]

## Compact description of region-wise packing information using scale factor

The following text modifications are proposed on top of OMAF WD for compact description of region-wise packing information using scale factor. The proposed changes to the text of the Region-wise packing structure section is shown highlighted compared to OMAF WD.

#### 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 this clause 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 informatively summarized below, while the normative 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, scale\_factor\_proj, scale\_factor\_packed),
  + 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, scale\_factor\_proj, scale\_factor\_packed) is informatively summarized below, while the normative semantics follow subsequently in this clause:

* If scale\_factor\_proj is set, then proj\_reg\_width\_scaled[i] \* scale\_factor\_proj, proj\_reg\_height\_scaled[i] \* scale\_factor\_proj, proj\_reg\_top\_scaled[i] \* scale\_factor\_proj, and proj\_reg\_left\_scaled[i] \* scale\_factor\_proj specify the width, height, top offset, and left offset, respectively, of the i-th projected region. Otherwise, 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.
* If scale\_factor\_packed is set, then packed\_reg\_width\_scaled[i] \* scale\_factor\_packed, packed\_reg\_height\_scaled[i] \* scale\_factor\_packed, packed\_reg\_top\_scaled[i] \* scale\_factor\_packed, and packed\_reg\_left\_scaled[i] \* scale\_factor\_packed specify the width, height, top offset, and left offset, respectively, of the i-th packed region. Otherwise, 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 informatively summarized below, while the normative 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.

**Error! Reference source not found.** 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.

…

This clause is organized as follows:

* The syntax and semantics of the rectangular region packing structure are specified in clauses **Error! Reference source not found.** and **Error! Reference source not found.**, respectively.
* The syntax and semantics of the guard band structure are specified in clauses **Error! Reference source not found.** and **Error! Reference source not found.**, respectively.
* The syntax and semantics of the region-wise packing structure are specified in clauses **Error! Reference source not found.** and **Error! Reference source not found.**, respectively.
* Clause 0 derives variables from syntax element values of the rectangular region packing, guard band, region-wise packing structures. Clause 0 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, scale\_factor\_proj, scale\_factor\_packed) {

if (scale\_factor\_proj) {

unsigned int(8) proj\_reg\_width\_scaled[i];  
 unsigned int(8) proj\_reg\_height\_scaled[i];  
 unsigned int(8) proj\_reg\_top\_scaled[i];  
 unsigned int(8) proj\_reg\_left\_scaled[i];

} else {  
 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;

if (scale\_factor\_packed) {

unsigned int(8) packed\_reg\_width\_scaled[i];  
 unsigned int(8) packed\_reg\_height\_scaled[i];  
 unsigned int(8) packed\_reg\_top\_scaled[i];  
 unsigned int(8) packed\_reg\_left\_scaled[i];

} else {  
 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 may 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 should be used for rendering.

proj\_reg\_width\_scaled[i], proj\_reg\_height\_scaled[i], proj\_reg\_top\_scaled[i], and proj\_reg\_left\_scaled[i] multiplied by scale\_factor\_proj specify the scaled 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\_scaled[i], proj\_reg\_height\_scaled[i], proj\_reg\_top\_scaled[i] and proj\_reg\_left\_scaled[i] multiplied by scale\_factor\_proj are indicated in relative projected picture sample units.

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: Clause **Error! Reference source not found.** 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 may partially or entirely overlap with each other.

packed\_reg\_width\_scaled[i], packed\_reg\_height\_scaled[i], packed\_reg\_top\_scaled[i], and packed\_reg\_left\_scaled[i] multiplied by scale\_factor\_packed specify the width, height, top offset, and 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\_scaled[i], packed\_reg\_height\_scaled[i], packed\_reg\_top\_scaled[i], and packed\_reg\_left\_scaled[i] multiplied by scale\_factor\_packed are indicated in relative packed picture sample units. packed\_reg\_width\_scaled[i], packed\_reg\_height\_scaled[i], packed\_reg\_top\_scaled[i], and packed\_reg\_left\_scaled[i] multiplied by scale\_factor\_packed shall represent integer horizontal and vertical coordinates of luma sample units within the decoded pictures.

#### 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 could 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 could 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 could 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;

unsigned int(1) scale\_factor\_flag;

~~bit(7) reserved = 0;~~

bit(6) reserved = 0;

if (scale\_factor\_flag) {

unsigned int(16) scale\_factor\_proj;

unsigned int(16) scale\_factor\_packed;

}  
 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, scale\_factor\_proj, scale\_factor\_packed);  
 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.

scale\_factor\_flag equal to 1 specifies that scale\_factor\_proj and scale\_factor\_packed are signaled. scale\_factor\_flag equal to 0 specifies that scale\_factor\_proj and scale\_factor\_packed are not signaled and inferred to be 0.

NOTE 2: Setting this flag equal to 1 together with setting at least one of scale\_factor\_proj and scale\_factor\_packed to a non-zero value, allows for more compact signalling of region-wise packing information.

scale\_factor\_proj specify the scale factor to be used to reconstruct the parameters for the projected regions. scale\_factor\_proj equal to 0 specifies that the values for the projected regions have not been scaled.

scale\_factor\_packed specify the scale factor to be used to reconstruct the parameters for the packed regions. scale\_factor\_proj equal to 0 specifies that the values for the packed regions have not been scaled.

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 **Error! Reference source not found.**.

RectRegionPacking(i, scale\_factor\_proj, scale\_factor\_packed) specifies the region-wise packing between the i-th packed region and the i-th projected region. The syntax and semantics of RectRegionPacking(i, scale\_factor\_proj, scale\_factor\_packed) are specified in clauses **Error! Reference source not found.** and **Error! Reference source not found.**, respectively.

GuardBand(i) specifies the guard bands for the i-th packed region. The syntax and semantics of GuardBand(i) are specified in clauses **Error! Reference source not found.** and **Error! Reference source not found.**, 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 should 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:
  + If scale\_factor\_packed > 0, the following applies:
    - PackedRegLeft[n] is set equal to packed\_reg\_left[n] \* scale\_factor\_packed.
    - PackedRegTop[n] is set equal to packed\_reg\_top[n] \* scale\_factor\_packed.
    - PackedRegWidth[n] is set equal to packed\_reg\_width[n] \* scale\_factor\_packed.
    - PackedRegHeight[n] is set equal to packed\_reg\_height[n] \* scale\_factor\_packed.
  + Otherwise 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].
  + If scale\_factor\_proj > 0, the following applies:
    - ProjRegLeft[n] is set equal to proj\_reg\_left[n] \* scale\_factor\_proj.
    - ProjRegTop[n] is set equal to proj\_reg\_top[n] \* scale\_factor\_proj.
    - ProjRegWidth[n] is set equal to proj\_reg\_width[n] \* scale\_factor\_proj.
    - ProjRegHeight[n] is set equal to proj\_reg\_height[n] \* scale\_factor\_proj.
  + Otherwise the following applies:
    - 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.
    - If scale\_factor\_packed > 0, the following applies:
      * PackedRegLeft[n] is set equal to packed\_reg\_left[nIdx] \* scale\_factor\_packed + packedLeftOffset.
      * PackedRegTop[n] is set equal to packed\_reg\_top[nIdx] \* scale\_factor\_packed + packedTopOffset.
      * PackedRegWidth[n] is set equal to packed\_reg\_width[nIdx] \* scale\_factor\_packed.
      * PackedRegHeight[n] is set equal to packed\_reg\_height[nIdx] \* scale\_factor\_packed.
    - Otherwise the following applies:
      * 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].
    - If scale\_factor\_proj > 0, the following applies:
      * ProjRegLeft[n] is set equal to proj\_reg\_left[nIdx] \* scale\_factor\_proj + projLeftOffset.
      * ProjRegTop[n] is set equal to proj\_reg\_top[nIdx] \* scale\_factor\_proj + projTopOffset.
      * ProjRegWidth[n] is set equal to proj\_reg\_width[nIdx] \* scale\_factor\_proj.
      * ProjRegHeight[n] is set equal to proj\_reg\_height[nIdx] \* scale\_factor\_proj.
    - Otherwise the following applies:
      * 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.

## Compact description of region-wise packing information for equal size regions and regions ordered in raster scan order

The following text modifications are proposed on top of OMAF WD for compact description of region-wise packing information for equal size regions and regions ordered in raster scan order. The proposed changes to the text of the Region-wise packing structure section is shown highlighted compared to OMAF WD.

#### 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 this clause 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 informatively summarized below, while the normative 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.
* When the projected regions have equal size, the flag proj\_reg\_equal\_size\_flag equal to 1 enables a compact description of the width and height of the projected regions.
* When the packed regions have equal size the flag packed\_reg\_equal\_size\_flag equal to 1 enables a compact description of the width and height of the packed regions.
* When the projected regions are ordered in raster scan order, the flag proj\_raster\_scan\_order\_flag equal to 1 enables a compact description of the top and left offsets of the projected regions.
* When the packed regions are ordered in raster scan order, the flag packed\_raster\_scan\_order\_flag equal to 1 enables a compact description of the top and left offsets of the packed regions.
* 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, proj\_reg\_equal\_size\_flag, packed\_reg\_equal\_size\_flag, proj\_raster\_scan\_order\_flag, packed\_raster\_scan\_order\_flag),
  + 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, proj\_reg\_equal\_size\_flag, packed\_reg\_equal\_size\_flag, proj\_raster\_scan\_order\_flag, packed\_raster\_scan\_order\_flag) is informatively summarized below, while the normative 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 informatively summarized below, while the normative 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.

**Error! Reference source not found.** 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.

…

This clause is organized as follows:

* The syntax and semantics of the rectangular region packing structure are specified in clauses **Error! Reference source not found.** and **Error! Reference source not found.**, respectively.
* The syntax and semantics of the guard band structure are specified in clauses **Error! Reference source not found.** and **Error! Reference source not found.**, respectively.
* The syntax and semantics of the region-wise packing structure are specified in clauses **Error! Reference source not found.** and **Error! Reference source not found.**, respectively.
* Clause 0 derives variables from syntax element values of the rectangular region packing, guard band, region-wise packing structures. Clause 0 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,

proj\_reg\_equal\_size\_flag,

packed\_reg\_equal\_size\_flag,

proj\_raster\_scan\_order\_flag,

packed\_raster\_scan\_order\_flag) {

if (proj\_reg\_equal\_size\_flag) {

if (i == 0) {

unsigned int(32) proj\_reg\_width[0];  
 unsigned int(32) proj\_reg\_height[0];

}

} else {  
 unsigned int(32) proj\_reg\_width[i];  
 unsigned int(32) proj\_reg\_height[i];

}

if (!proj\_raster\_scan\_order\_flag) {  
 unsigned int(32) proj\_reg\_top[i];  
 unsigned int(32) proj\_reg\_left[i];

}  
 unsigned int(3) transform\_type[i];  
 bit(5) reserved = 0;

if (packed\_reg\_equal\_size\_flag) {

if (i == 0) {

unsigned int(16) packed\_reg\_width[0];  
 unsigned int(16) packed\_reg\_height[0];

}

} else {   
 unsigned int(16) packed\_reg\_width[i];  
 unsigned int(16) packed\_reg\_height[i];

}

if (!packed\_raster\_scan\_order\_flag) {  
 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 may 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 should 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: Clause **Error! Reference source not found.** 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 may 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 could 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 could 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 could 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;

unsigned int(1) proj\_reg\_equal\_size\_flag;

unsigned int(1) packed\_reg\_equal\_size\_flag;

unsigned int(1) proj\_raster\_scan\_order\_flag;

unsigned int(1) packed\_raster\_scan\_order\_flag;

~~bit(7) reserved = 0;~~

bit(3) 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, proj\_reg\_equal\_size\_flag,

packed\_reg\_equal\_size\_flag,

proj\_raster\_scan\_order\_flag,

packed\_raster\_scan\_order\_flag);  
  
 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.

proj\_reg\_equal\_size\_flag equal to 1 specifies that all regions of the projected picture have the same width and height. The regions in the left column of the picture may have a shorter width than regions in other columns and the regions at the bottom row of the picture may have a shorter height than regions in other rows. proj\_reg\_equal\_size\_flag equal to 0 specifies that all regions of the projected picture may not have equal size.

NOTE 2: For projected regions where the regions have equal size, setting this flag equal to 1 allows more compact signalling of region-wise packing information by only signalling the size for the first region.

packed\_reg\_equal\_size\_flag equal to 1 specifies that all regions of the packed picture have the same width and height. The regions in the left column of the picture may have a shorter width than regions in other columns and the regions at the bottom row of the picture may have a shorter height than regions in other rows. packed\_reg\_equal\_size\_flag equal to 0 specifies that all regions of the packed picture may not have equal size.

NOTE 3: For packed regions where the regions have equal size, setting this flag equal to 1 allows more compact signalling of region-wise packing information by only signalling the size for the first region.

proj\_raster\_scan\_order\_flag equal to 1 specifies that all regions of the projected picture are ordered in raster scan order. For stereoscopic frame packing formats the regions are ordered in raster scan order for each stereo view. proj\_reg\_equal\_size\_flag equal to 0 specifies that all regions of the projected picture may not be orderered in raster scan order.

NOTE 4: For projected regions where the regions are ordered in raster scan order, setting this flag equal to 1 allows more compact signalling of region-wise packing information by deriving the region offsets from the region width and region height.

packed\_raster\_scan\_order\_flag equal to 1 specifies that all regions of the packed picture are ordered in raster scan order. For stereoscopic frame packing formats the regions are ordered in raster scan order for each stereo view. packed\_reg\_equal\_size\_flag equal to 0 specifies that all regions of the packed picture may not be ordered in raster scan order.

NOTE 5: For packed regions where the regions are ordered in raster scan order, setting this flag equal to 1 allows more compact signalling of region-wise packing information by deriving the region offsets from the region width and region height.

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 4: 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 **Error! Reference source not found.**.

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 clauses **Error! Reference source not found.** and **Error! Reference source not found.**, respectively.

GuardBand(i) specifies the guard bands for the i-th packed region. The syntax and semantics of GuardBand(i) are specified in clauses **Error! Reference source not found.** and **Error! Reference source not found.**, 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 should 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:
  + If packed\_raster\_scan\_order\_flag is equal to 1, the following applies:
    - PackedRegLeft[n] is set equal to the x-position of the first (x, y)-point in raster scan order of the packed picture not already occupied by a region.
    - PackedRegTop[n] is set equal to the y-position of the first (x, y)-point in raster scan order of the packed picture not already occupied by a region.
  + Otherwise the following applies:
    - PackedRegLeft[n] is set equal to packed\_reg\_left[n].
    - PackedRegTop[n] is set equal to packed\_reg\_top[n].
  + If packed\_reg\_equal\_size\_flag is equal to 1, the following applies:
    - If packed\_reg\_width[0] > packed\_picture\_width/HorDiv1 - PackedRegLeft[n] then PackedRegWidth[n] is set equal to packed\_picture\_width/HorDiv1 - PackedRegLeft[n]. Otherwise, PackedRegWidth[n] is set equal to packed\_reg\_width[0].
    - If packed\_reg\_height[0] > packed\_picture\_height/VerDiv1 - PackedRegTop[n] then PackedRegHeight[n] is set equal to packed\_picture\_height/VerDiv1 - PackedRegTop[n]. Otherwise, PackedRegHeight[n] is set equal to packed\_reg\_height[0].
  + Otherwise the following applies:
    - PackedRegWidth[n] is set equal to packed\_reg\_width[n].
    - PackedRegHeight[n] is set equal to packed\_reg\_height[n].
  + If proj\_raster\_scan\_order\_flag is equal to 1, the following applies:
    - ProjRegLeft[n] is set equal to the x-position of the first (x, y)-point in raster scan order of the projected picture not already occupied by a region.
    - ProjRegTop[n] is set equal to the y-position of the first (x, y)-point in raster scan order of the projected picture not already occupied by a region.
  + Otherwise the following applies:
    - ProjRegLeft[n] is set equal to proj\_reg\_left[n].
    - ProjRegTop[n] is set equal to proj\_reg\_top[n].
  + If proj\_reg\_equal\_size\_flag is equal to 1, the following applies:
    - If proj\_reg\_width[0] > proj\_picture\_width/HorDiv1 - ProjRegLeft[n] then ProjRegWidth[n] is set equal to proj\_picture\_width/HorDiv1 - ProjRegLeft[n]. Otherwise, ProjRegWidth[n] is set equal to proj\_reg\_width[0].
    - If proj\_reg\_height[0] > proj\_picture\_height/VerDiv1 - ProjRegTop[n] then ProjRegHeight[n] is set equal to proj\_picture\_height/VerDiv1 - ProjRegTop[n]. Otherwise, ProjRegHeight[n] is set equal to proj\_reg\_height[0].
  + Otherwise the following applies:
    - 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.
    - If packed\_raster\_scan\_order\_flag is equal to 1, the following applies:
      * PackedRegLeft[n] is set equal to the x-position + packedLeftOffset of the first (x, y)-point in raster scan order of the packed picture not already occupied by a region.
      * PackedRegTop[n] is set equal to the y-position + packedTopOffset of the first (x, y)-point in raster scan order of the packed picture not already occupied by a region.
    - Otherwise the following applies:
      * PackedRegLeft[n] is set equal to packed\_reg\_left[nIdx] + packedLeftOffset.
      * PackedRegTop[n] is set equal to packed\_reg\_top[nIdx] + packedTopOffset.
    - If packed\_reg\_equal\_size\_flag is equal to 1, the following applies:
      * If packed\_reg\_width[0] > packed\_picture\_width/HorDiv1 + packedLeftOffset - PackedRegLeft[n] then PackedRegWidth[n] is set equal to packed\_picture\_width/HorDiv1 + packedLeftOffset - PackedRegLeft[n]. Otherwise, PackedRegWidth[n] is set equal to packed\_reg\_width[0].
      * If packed\_reg\_height[0] > packed\_picture\_height/VerDiv1 + packedTopOffset - PackedRegTop[n] then PackedRegHeight[n] is set equal to packed\_picture\_height/VerDiv1 + packedTopOffset - PackedRegTop[n]. Otherwise, PackedRegHeight[n] is set equal to packed\_reg\_height[0].
    - Otherwise the following applies:
      * PackedRegWidth[n] is set equal to packed\_reg\_width[nIdx].
      * PackedRegHeight[n] is set equal to packed\_reg\_height[nIdx].
    - If proj\_raster\_scan\_order\_flag is equal to 1, the following applies:
      * ProjRegLeft[n] is set equal to the x-position + projLeftOffset of the first (x, y)-point in raster scan order of the projected picture not already occupied by a region.
      * ProjRegTop[n] is set equal to the y-position + projTopOffset of the first (x, y)-point in raster scan order of the projected picture not already occupied by a region.
    - Otherwise the following applies:
      * ProjRegLeft[n] is set equal to proj\_reg\_left[nIdx] + projLeftOffset.
      * ProjRegTop[n] is set equal to proj\_reg\_top[nIdx] + projTopOffset.
    - If proj\_reg\_equal\_size\_flag is equal to 1, the following applies:
      * If proj\_reg\_width[0] > proj\_picture\_width/HorDiv1 + projLeftOffset - ProjRegLeft[n] then ProjRegWidth[n] is set equal to proj\_picture\_width/HorDiv1 + projLeftOffset - ProjRegLeft[n]. Otherwise, ProjRegWidth[n] is set equal to proj\_reg\_width[0].
      * If proj\_reg\_height[0] > proj\_picture\_height/VerDiv1 + projTopOffset - ProjRegTop[n] then ProjRegHeight[n] is set equal to proj\_picture\_height/VerDiv1 + projTopOffset - ProjRegTop[n]. Otherwise, ProjRegHeight[n] is set equal to proj\_reg\_height[0].
    - Otherwise the following applies:
      * 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.

# Extending the 'cdtg' track reference type to support referring to both tracks and items

Currently, there is no existing mechanisms for association of an image item to a timed metadata track.

If the file format group agrees with the proposed update to the semantics of track\_IDs in TracReferenceBox, including the part of track\_IDs equal to 0, the 'cdtg' track reference type would be extended to support referring to both tracks and items.

The extensions to the semantics of track\_IDs in TracReferenceBox mentioned above are as follows:

track\_IDsis an array of integers providing the track or item identifiers of the referenced tracks or items or track\_group\_id values of the referenced track groups. Each value track\_IDs[i], where i is a valid index to the track\_IDs[] array,is an integer that provides a reference from the containing track to the track with track\_id equal to track\_IDs[i], to the item with item\_id equal to track\_IDs[i]or to the track group with both track\_group\_id equal to track\_IDs[i] and (flags & 1) of TrackGroupTypeBox equal to 1. When a track\_group\_id value is referenced, the track reference applies to each track of the referenced track group individually unless stated otherwise in the semantics of particular track reference types. The value 0 shall not be present. In the array there shall be no duplicated value; however, a track\_id may appear in the array and also be a member of one or more track groups for which the track\_group\_ids appear in the array. This means that in forming the list of tracks, after replacing track\_group\_ids by the track\_ids of the tracks in those groups, there might be duplicate track\_ids.

When an ID value equal to zero (0) is present in the track\_IDs array, it represents a placeholder for a non-zero track ID value or a non-zero item ID that is to be resolved at the time the overlay is performed; this allows the use case of dynamic overlay.

# Support of transparent background in OMAF

## Summary

It is proposed to add the changes shown in text below to the OMAF v2 WD to enable support for overlays with transparent background:

## Proposed specification text

It is proposed to include the following text changes to the next revision of the OMAF v2 WD. The proposed changes are shown in red compared to OMAF WD4.

**3.1.4 background layer**

layer on which an *overlay* is superimposed. Background layer can be either *background visual media* or a *transparent background layer*

**3.1.5 background visual media**

piece of *visual media* on which an *overlay* is superimposed

…

**3.1.50 transparent background layer**

*background layer* intended for supporting OMAF output with transparent background. The transparent background layer is not signaled in a track or image item.

…

#### 4.4.4.3 Restricted scheme types

The restricted scheme types specified in this document are listed in Table 4.3.

**Table 4.3 – Restricted scheme types specified in this document**

|  |  |  |
| --- | --- | --- |
| **Restricted scheme type** | **Clause in this document** | **Informative description** |
| … | … | … |
| ecov | 0 | Like 'ercm' but overlay and viewing space information may additionally be present. |
| ectb | 0 | Like 'ecov' but overlays may be superimposed on top of a transparent background. |

…

#### 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 and ViewingSpaceBox. The value of version of OverlayConfigBox (when present) and ViewingSpaceBox (when present) shall be equal to 0.

#### 7.6.1.7 Equirectangular or cubemap projected video with overlays over transparent background ('ectb')

When scheme\_type is equal to 'ectb' in an instance of CompatibleSchemeTypeBox in the RestrictedSchemeInfoBox, the track conforms to the constraints of scheme\_type equal to 'ecov' except that the value of background\_layer\_type in OverlayAndBackgroundGroupingBox may be equal to 1.

**7.13.1 Overlay structure**

**7.13.1.1 Definition**

OverlayStruct specifies the overlay related metadata per each overlay.

**7.13.1.2 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();  
}

…

…

* 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~~background layer.

…

**7.13.2.2 Viewport-relative 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~~background layer.

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~~background layer.

…

unit\_sphere\_distance\_in\_mm specifies a distance, in millimeters, corresponding to the radius of the unit sphere. The value should be used 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. When present multiple times for the same ~~media~~background layer, the value of all instances of unit\_sphere\_distance\_in\_mm shall be identical. The presence of unit\_sphere\_distance\_in\_mm is optional.

**7.13.6.2 Grouping of overlays and background visual media that are intended to be presented together**

##### 7.13.6.2.1 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.

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 could 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.

If the i-th entity in the 'ovbg' entity group includes background visual media, background\_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 could 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.

An 'ovbg' entity group with background\_layer\_type equal to 0 shall contain either a background visual media track or a background image item but not both. Additionally, any two background visual media tracks in the same 'ovbg' entity group shall be alternatives to each other, indicated by the same value of alternate\_group in their TrackHeaderBox, or shall belong to the same 2D spatial relationship track group. Any two background visual image items in the same 'ovbg' entity group shall belong to the same 'altr' entity 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.

An 'ovbg' entity group with background\_layer\_type equal to 1 shall not contain any background visual media tracks nor any background image items.

##### 7.13.6.2.1 Syntax

aligned(8) class OverlayAndBackgroundGroupingBox(version, flags)   
extends EntityToGroupBox('ovbg', version, flags) {  
 for(i=0; i<num\_entities\_in\_group; i++) {  
 bit(6) reserved = 0;  
 unsigned int(1) overlay\_flag[i];  
 unsigned int(1) background\_flag[i];  
 }  
}

##### 7.13.6.2.1 Semantics

background\_layer\_type specifies the type of the background layer on which the overlays in the entity group are superimposed. background\_layer\_type equal to 0 specifies that the background layer is background visual media. background\_layer\_type equal to 1 specifies that the background layer is transparent background layer. For tracks and image items not pertaining to a BackgroundLayerTypeGroupingBox the background layer is inferred to be background visual media. Values of background\_layer\_type greater than 1 are reserved.

NOTE: background\_layer\_type equal to 1 is intended for OMAF output with transparent background. When the background layer is transparent background layer then the background layer is not signaled in a track or image item.

**7.13.8.2 Sphere-relative overlay rendering procedure**

1. Setup the VR scene geometry by creating the sphere and placing the rendering camera in the centre of the sphere (depending on whether the content is stereo or mono, the rendering camera has to correspondingly be mono or stereo).

…

1. If background\_layer\_type is equal to 0, r~~R~~ender the background omnidirectional visual media.

…

# OMAF late binding DASH streaming application model

The diagram below shows current understanding regarding division of functionality between OMAF client and DASH client.

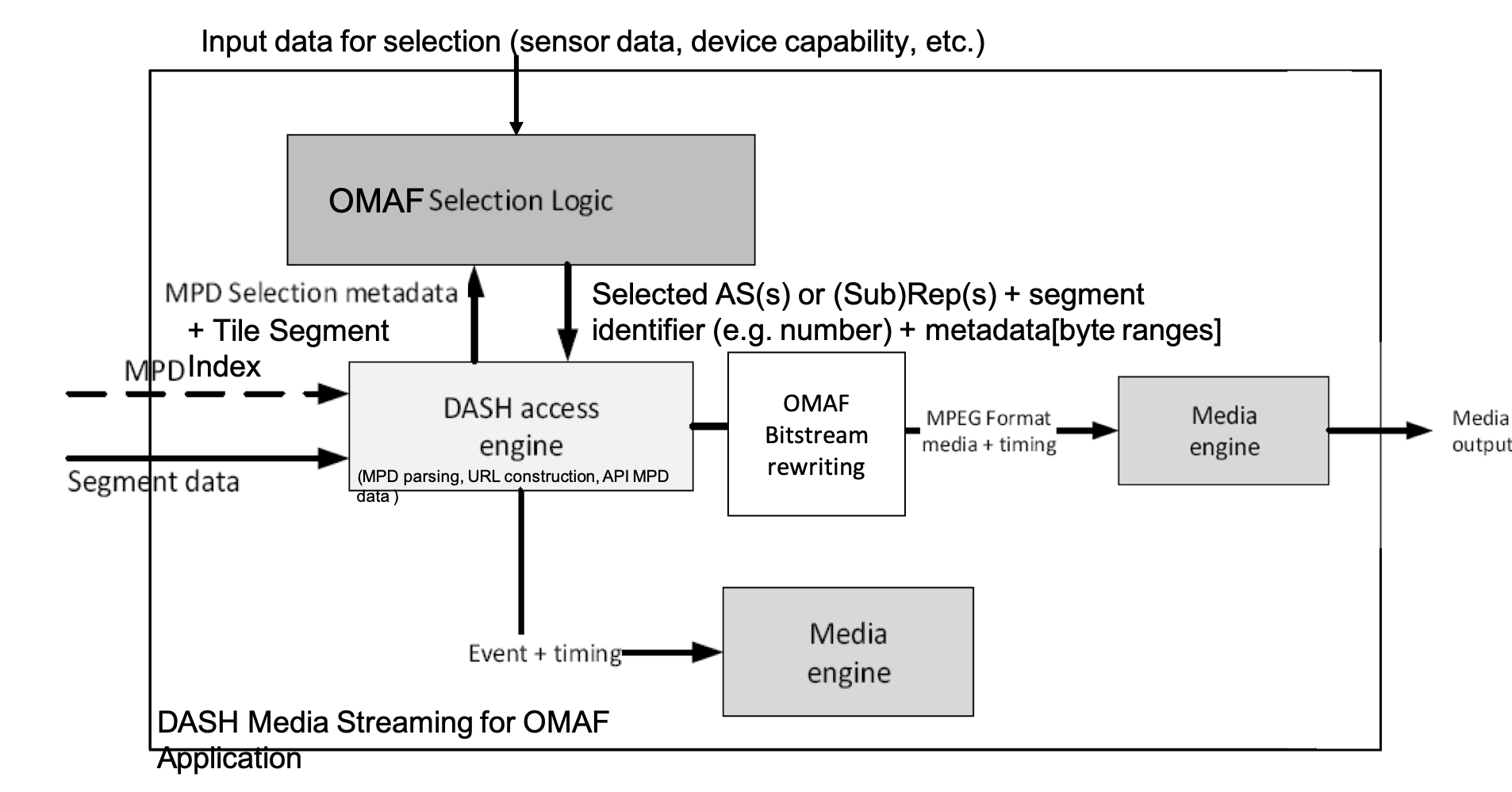


Figure 1: OMAF late binding DASH streaming application model

# Recommended viewport signaling

Updates are proposed to the DASH signaling for the recommended viewport. It is asserted that this enables a player to select a higher quality and more uniform quality media track representations. This latter aspect is asserted to be especially important for watching content on 2D displays.

## Proposal

A **SupplementalProperty** with a @schemeIdUri attribute equal to "urn:mpeg:mpegI:omaf:2018:recv" is referred to as an recommended viewport descriptor.

One recommended viewport descriptor may be present at a representation level. The presence of the recommended viewport descriptor indicates that the representation is optimized for rendering recommended viewport

~~track~~indicated by referenced recommended viewport timed metadata representation (e.g. the omnidirectional video in the representation is encoded such that the quality of the sphere region corresponding to the recommended viewport is always higher than other region).

.

One and only one recommended viewport timed metadata representation shall be associated with the representation containing the recommended viewport descriptor.

The @value descriptor shall not be present.

# Richer interaction signalling for overlays

The proposal introduces a mechanism that allows to define much richer overlays by extending their interactivity capability.

## Signalling in ISOBMFF

In order to implement the proposed solution here, the following changes, highlighted in yellow, are proposed for section 7.13.2.9 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) viewpoint\_switching\_flag;

unsigned int(1) default\_action\_flag;

unsigned int(1) secondary\_action\_flag;

if (viewpoint\_switching\_flag)

{

unsigned int(16) destination\_viewpoint\_id;

}

if (default\_action\_flag) { unsigned int(8) action\_id; }

if (secondary\_action\_flag) { unsigned int(8) action\_id; }

bit(6) reserved = 0;

}

With the following additional semantics:

viewpoint\_switching\_flag equal to 1 specifies that a viewpoint switching to the viewpoint identified by destination\_viewpoint\_id shall be initiated when interacting with the media overlay.

default\_action\_flag equal to 1 specifies that a default action is set for the overlay.

secondary\_action\_flag equal to 1 specifies that a secondary action is set for the overlay.

action\_id specifies the action to use as defined in the following table:

|  |  |
| --- | --- |
| action\_id | Action |
| 0 | Change position of overlay. Note: change\_position\_flag shall be set to 1. |
| 1 | Change depth of overlay. Note: change\_depth\_flag shall be set to 1. |
| 2 | Switch overlay on/off. Note: switch\_on\_off\_flag shall be set to 1. |
| 3 | Change opacity of overlay. Note: change\_opacity\_flag shall be set to 1. |
| 4 | Resize overlay. Note: resize\_flag shall be set to 1. |
| 5 | Rotate overlay. Note: rotation\_flag shall be set to 1. |
| 6 | Change the source of the overlay. Note: source\_switching\_flag shall be set to 1. |
| 7 | Initiate a viewpoint switching. Note: viewpoint\_switching\_flag shall be set to 1. |

# Signalling for a track/ image item containing both background and overlay

Summary of this proposal is to add indication of omnidirectional video information for the overlay visual media when packedencapsulation*.*

The newly defined OmniVideoInfoStruct that indicates omnidirectional video information for the overlay visual media encapsulated into same track or image item as the background visual media is added to the overlay control structures.

## Proposed updated on the specification text

The proposed update on the specification text in OMAF 2nd edition CD is following. The updated portion is highlighted in yellow.

**7.13.2 Overlay control structures**

**7.13.2.1 Definition**

Controls with bit indices 0 to 2 specify the rendering type of the overlay. One and exactly one of overlay\_control\_flag[i] with i in the range of 0 to 2, inclusive, in each SingleOverlayStruct shall be equal to 1.

Controls with bit indices 3 and 4 specify the source of the overlay content. At most one of overlay\_control\_flag[3] and overlay\_control\_flag[4] shall be equal to 1 in SingleOverlayStruct. If neither overlay\_control\_flag[3] nor overlay\_control\_flag[4] in SingleOverlayStruct is equal to 1, the overlay is the entire associated decoded picture.

Control with bit index 12 specifies the omnidirectional video information of the overlay visual media. When the overlay visual media is included in the same track or image item with background visual media, overlay\_control\_flag[12] shall be equal to 1 if the overlay visual media is omnidirectional video, overlay\_control\_flag[12] shall be equal to 0 if the overlay visual media is 2D video.

NOTE: When the overlay visual media is included in the same track or image item with background visual media, the omnidirectional video information directly signaled in the ProjectedOmniVideoBox or ItemPropertyContainerBox is applied to the background visual media.

**Table 7.4 – Overlay control structures**

|  |  |  |
| --- | --- | --- |
| **Bit index** | **Clause in this document** | **Description** |
| 0 | 7.13.2.2 | Parameters for viewport-relative overlay |
| 1 | 7.13.2.3 | Parameters for sphere-relative projected omnidirectional overlay |
| 2 | 7.13.2.4 | Parameters for sphere-relative 2D overlay |
| 3 | 7.13.2.5 | Source region for the overlay.  Indicates the region within the decoded picture that is used as the content of the overlay. |
| 4 | 7.13.2.6 | Recommended viewport overlay.  Indicates the recommended viewport track whose recommended viewports are used as the content of the overlay. |
| 5 | 7.13.2.7 | Overlay layering order |
| 6 | 7.13.2.8 | Overlay opacity |
| 7 | 7.13.2.9 | Controls for user interaction |
| 8 | 7.13.2.10 | Overlay label |
| 9 | 7.13.2.11 | Overlay priority |
| 10 | 7.13.2.12 | Associated sphere region |
| 11 | 7.13.2.13 | Overlay alpha composition |
| 12 | 7.13.2.14 | Omnidirectional video information for overlay |

LastControlIdx is set equal to ~~11~~12.

…

**7.13.2.14 Omnidirectional video information for overlay**

aligned(8) class OmniVideoInfoStruct() {  
 unsigned int(1) rwpk\_struct\_flag;  
 bit(7) reserved = 0;  
 ProjectionFormatStruct();  
 if(rwpk\_struct\_flag)  
 RegionWisePackingStruct();  
}

rwpk\_struct\_flag equal to 1 specifies that the RegiongWisePackingStruct() is present. rwpk\_struct\_flag equal to 0 specifies that the RegiongWisePackingStruct() is not present.

ProjectionFormatStruct() is identical to the definition in clause 7.5.2, and provides information on the projection format for the overlay source region.

RegionWisePackingStruct() is identical to the definition in clause 7.5.3. The presence of RegionWisePackingStruct() indicates that the overlay source region is packed region-wise and require unpacking prior to rendering, according to the region-wise packing process information as indicated.

# Viewpoint switching behavior flag

## Motivation for different viewpoint switching behavior

OMAFv2 supports for multiple viewpoints is versatile. Viewpoints represent different viewing position to the same scenes or different scenes. There can be different expectations for user experience when switching from one viewpoint to another. For some content, an immediate response by stopping the playback of the current viewpoint immediately, indicating on the display that a viewpoint switch is progressing e.g. by showing a transition effect, and switching to the new viewpoint as soon as possible, is preferred. For other type of content, ensuring content continuity over the switch and hence the need to keep playing the current viewpoint as long as needed, is preferred.

For example:

1. For sports content, it is more important that the game is visible continuously even if there is perceived latency in terms of switching the viewpoint. Other such examples could be viewpoints which are part of the same group or viewpoints with common visual scene.
2. For multiple viewpoint content where the user is browsing through different viewpoints which are not part of the same viewpoint group, it would be more intuitive if the content switches as soon as possible. An end-user perceives an immediate switch as a responsive reaction to his/her initiation for the viewpoint switch. This is analogous to fast channel surfing.

## Need for changes to the specification

However, there is no clear specification about what should happen with the current viewpoint data during switching, i.e. whether the player should continue playing it or not. In practice, there is typically a delay in the order of hundreds of milliseconds to a few seconds before the playback of the new viewpoint can start.

Different player implementations might make different implementation choices due to the absence of clear specification to guide the behavior. As a result, there is a risk of inconsistent player behavior and unexpected user experience, if a player chooses to delay the switching instances to ensure continuous playback (e.g., if content is available for current viewpoint but not yet for destination viewpoint) or chooses to switch immediately to a new viewpoint with the associated risk of not having the content available to playback from the next playout sample.

## Proposal

The proposal is described in two parts. The first part proposes a new flag in the ViewpointTimelineSwitchStruct(). The second part relates to behaviour based on the new signaling in the ISOBMFF file format.

### *Addition of new flag to Viewpoint switching list structure (7.12.1.7.1)*

aligned(8) class ViewpointSwitchingListStruct() {  
 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;  
 unsigned int(1) viewpoint\_switch\_type\_flag;  
 bit(2~~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)  
 utf8string transition\_video\_URL;  
 }  
 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();  
 }  
 }  
}

### *Semantics*

If the switch\_type\_flag is set to 1, the player should switch to the new viewpoint immediately.

*Note: The switch may take place with a period of black screen or a transition effect, if a transition effect is configured for the switch.*

If the switch\_type\_flag is set to 0, the player should continue playing the current viewpoint until content for the destination viewpoint is ready to be rendered.

*Note: The player may show some indication that the switch is in progress.*

### *Addition to Expected OMAF player behavior Viewpoint switching (G.7.2.3)*

If the viewpoint\_switch\_type flag is equal to 1, the current viewpoint content playback stops immediately and the playback switches to rendering content from the next sample of the destination viewpoint. In absence of playable content from the destination viewpoint, the switching interval shows a black screen or a transition effect, if a transition effect is configured for the switch and supported by the player. If the viewpoint\_switch\_type\_flag is equal to 0, the current viewpoint content playback continues until playable content is available for the destination viewpoint. The player is expected to show an indication that the switch is in progress.

# OMAF Tile Track Layout

## Introduction

An OMAF base track could use different partitionings to slices and tiles for the same layout of OMAF tile tracks, thus resulting into different mapping of the OMAF tile tracks to 2D positions within the decoded picture. For example, in the examples of Figure D.13 of OMAF 2nd Ed., both row-major and column-major orders of OMAF tile tracks would be possible. Presently, an OMAF player must parse the slice segment headers of the OMAF base track to figure out the 2D position of each OMAF tile track within the decoded picture.

To simplify the operation of the OMAF player and to avoid HEVC bitstream parsing in the OMAF player, it is proposed to specify the signalling of the layout of the OMAF tile tracks within the decoded picture resulting from decoding the OMAF base track. The signalling is proposed to be mandatory with the 'erc2' scheme type and consequently in the simple tiling profile.

## Proposed specification text

### Tiling layout box

**7.6.9 Tiling layout box**

**7.6.9.1 Definition**

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

TilingLayoutBox provides the 2D layout of the referenced OMAF tile tracks within each sample of the OMAF base track that references a sample entry carrying this box. The position, width, and height of each OMAF tile track within the decoded picture resulting from a sample of the OMAF tile base track is given in the order that the sample of the OMAF tile base track references the OMAF tile tracks.

**7.6.9.2 Syntax**

aligned(8) class TilingLayoutBox() extends FullBox ('tilo', 0, 0) {  
 unsigned int(16) num\_tiles\_minus1;  
 for(i=0; i <= num\_tiles\_minus1; i++) {  
 unsigned int(16) tile\_top[i];  
 unsigned int(16) tile\_left[i];  
 unsigned int(16) tile\_width[i];  
 unsigned int(16) tile\_height[i];  
 }  
}

**7.6.9.3 Semantics**

num\_tiles\_minus1 plus 1 specifies the number of tiles referenced by each sample of the OMAF base track that references a sample entry carrying this box.

tile\_top[i] specifies the top offset of the tile from the top boundary of the decoded picture in luma sample rows with value 0 being the top-most luma sample row of the decoded picture.

tile\_left[i] specifies the left offset of the tile from the left boundary of the decoded picture in luma sample columns with value 0 being the left-most luma sample column of the decoded picture.

tile\_width[i] specifies the width of the tile in luma samples.

tile\_height[i] specifies the height of the tile in luma samples.

### 'erc2' scheme type (clause 7.6.1.7)

*Add the following:*

* TilingLayoutBox shall be present in the track containing the 'erc2' scheme type.

*Add the required version of the TilingLayoutBox:*

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

*Add TilingLayoutBox among the allowed boxes:*

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

# Tile-based Viewport Dependent Streaming

This section proposes an additional function for existing tile-based viewport-dependent profiles.

## Introduction

There are tile-based viewport dependent profiles, such as the VVC-based simple tiling OMAF video profile, in OMAF 3rd ed. These profiles can be used for viewport adaptive streaming using DASH.

An example of tile-based streaming is that several versions of the omnidirectional video content are coded at the same resolution but different qualities and bitrate in D.4 “Equal-resolution OMAF tile bitstreams merged to one bitstream” of OMAF 3rd ed.

This approach generates content in the following steps:

* Encoding: Several bitstreams of the same omnidirectional source content are encoded at the same resolution but different qualities and bitrates.
* Encapsulation of coded data: OMAF tile tracks and an OMAF base track are formed.

An OMAF player performs the following operations:

* The OMAF player chooses which version of each OMAF tile track is received, for example based on region-wise quality ranking information and the viewing orientation.
* The same OMAF base track suffices for combining OMAF tile tracks originating from different bitstreams. The OMAF player parses the OMAF base track to reconstruct a bitstream from the OMAF tile tracks.
* The reconstructed bitstream is decoded with a conforming decoder.

The tile-based streaming has the following characteristics.

1. Lower encoding cost for content authoring than without tile tracks  
   If viewport-dependent adaptation without using tile tracks, many variations of the viewport-dependent bitstream have to be encoded.
2. OMAF player needs implementations of quality selection per tile and merge for decoding. The amount of data increases compared to viewport-dependent adaptation without using tile tracks because of per-tile encapsulation.

## Proposal

It proposes a method to improve the disadvantages of the OMAF Player and to keep the advantages of encoding cost. This method allows the server to handle some of the functions of the OMAF player. And it proposes additional functions for existing tile-based view-dependent profiles.

To take advantage of encoding, the content authoring is the same as tile-based streaming, but the OMAF Player doesn’t handle the functions of quality selection of tile tracks and merging bitstreams from tile tracks. The sever handles these functions.

Figure D.5 in OMAF 3rd ed **Error! Reference source not found.**, which is an example using a VVC-based simple tilting OMAF video profile, is used to explain. As shown in Figure 1, it is a content authoring part and OMAF player processing part, and the yellow part in OMAF player is functions, quality selection and merging bitstream.

A picture containing graphical user interface

Description automatically generated

Figure 1 Figure D.5 in OMAF 3rd ed

The proposed process flow is shown in Figure 2. It adds server-side processing. The server-side processing has the functions that are in yellow part of Figure 1 and encapsulation of merged data which is a viewport-dependent mixed-quality bitstream. The server delivers a file that is encapsulated merged data to OMAF player.

This file can be played in a OMAF Player that does not support OMAF tiling track.A picture containing graphical user interface

Description automatically generated

Figure 2 Proposed process

In this proposed process flow, the OMAF player's viewport information is required on the server-side processing. It proposes an additional function for this requirement.

This function indicates the need to send the viewport information to the server in the metadata of the DASH MPD and specifies the method of sending the viewport information.

The proposal text is below:

x.x.x Signalling of server-side viewport adaptation information

A **EssentialProperty** element with a @schemeIdUri attribute equal to "urn:mpeg:mpegI:omaf:2022:serverSideAdaptation" is referred to as a server-side viewport adaptation (SSVA) descriptor.

When SSVA descriptor is exist in AdaptationSet or Representation, the query parameter which indicate the SphereRegion is added to the segment file URL. The values of indication of Sphere Region are centre\_azimuth, centre\_elevation, centre\_tilt, azimuth\_range, and elevation\_range.

Descriptions of these values are same as the values of cc.coverageInfo.

ex. http://hoge.com/vvc.mp4?centre\_azimuth=9830400&centre\_elevation=3932160&centre\_tilt=0& azimuth\_range=3932160&elevation\_range=3932160

----