International Standard

ISO/IEC 23090-7

First edition
2022-11

AMENDMENT 1
2024-12

Information technology — Coded representation of immersive media —

Part 7: Immersive media metadata

AMENDMENT 1: Common metadata for immersive media

Reference number
ISO/IEC 23090-7:2022/Amd. 1:2024(en)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Foreword

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

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

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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

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

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

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

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Information technology — Coded representation of immersive media —

Part 7: Immersive media metadata

AMENDMENT 1: Common metadata for immersive media

Normative references

Add the following reference to the end of the reference list:

IEEE 754-2019. *IEEE Standard for Floating-Point Arithmetic*

Introduction

Replace:

— Clause 5 describes common metadata applicable to immersive media. This includes reference co-ordinate system related metadata and other common metadata syntax and semantics.

— Clause 6 describes metadata that applies to video and images. This includes projection formats and region-wise packing metadata metadata which applies to video and images.

with:

— Clause 5 describes common metadata applicable to immersive media, with omnidirectional media in particular. This includes reference co-ordinate system related metadata and other common metadata syntax and semantics.

— Clause 6 describes metadata that applies to video and images, with omnidirectional media in particular. This includes projection formats and region-wise packing metadata metadata which applies to video and images.

— Clause 7 describes common metadata applicable to immersive media, with Visual Volumetric Video-based Coding (V3C) and Video-based Point Cloud Compression (V-PCC) in particular. This includes extrinsic camera information, intrinsic camera information, and other 3D common metadata syntax and semantics.

— Appendices A and B on Annotations of non-timed visual volumetric data and G-PCC data.

Add the following paragraphs after the document organization list

This document follows the following guiding principles:

1) Common metadata and their data structures shall be defined for both 3DoF and 6DoF immersive content, separately as well as jointly, in order to be used for applications that are either specific to separate 3DoF and 6DoF immersive content or general to mixed 3DoF and 6DoF immersive content.

2) Basic and common data structures are defined for simple metadata, and extend and enhanced data structures are defined as extensions of basic and common metadata (e.g., viewport is an extension of viewpoint)

3) Metadata structures shall be defined in a way to allow their encapsulation in ISOBMFF:

— Static: extension of containing boxes

— Dynamic: timed metadata tracks

Clause 7

Add the following new Clause 7 after Clause 6.

7 Common Metadata for Immersive Media

7.1 Vector3

Dimensions, positions, sizes for 3D immersive media can be defined using the following 3D vector data structure.

7.1.1 Syntax

```
aligned(8) class Vector3(unsigned char precision_bytes_minus1) {
   signed int((precision_bytes_minus1+1)*8) x;
   signed int((precision_bytes_minus1+1)*8) y;
   signed int((precision_bytes_minus1+1)*8) z;
}
```

7.1.2 Semantics

`precision_bytes_minus1`: Plus 1, specifies the precision of Vector3 components in bytes. Valid values are in the range from `[0, 3]`.

`x`, `y` and `z`: specify the x, y, and z coordinate values, respectively, of a 3D point in the Cartesian coordinate system

7.2 Scaling

Scaling in 3-dimension is defined using the following data structure:

7.2.1 Syntax

```
aligned(8) class 3DScaling(unsigned char precision_bytes_minus1) {
   Vector3 scale(precision_bytes_minus1);
}
```

7.2.2 Semantics

`precision_bytes_minus1`: Plus 1, specifies the precision of `scale` components in bytes. Valid values are in the range from `[0, 3]`.

`scale.x`, `scale.y`, and `scale.z` indicate the scaling extension in the Cartesian coordinates along the x, y, and z axes, respectively, relative to the origin `(0,0,0)`.

7.3 Extrinsic Camera Information

Extrinsic camera information is defined using the following data structure.

7.3.1 Syntax

```
class CameraExtrinsics(unsigned char abs_flag, unsigned char mode, unsigned char pos_bytes_
minus1, unsigned char pos_unit, unsigned char quat_bytes_minus1, unsigned char quat_den_bits_
minus1) {
   if(mode & 0x1) {
      signed int((pos_bytes_minus1+1)*8) pos_x;
```

```
    }
    if(mode & 0x2) {
       signed int((pos_bytes_minus1+1)*8) pos_y;
    }
    if(mode & 0x4) {
       signed int((pos_bytes_minus1+1)*8) pos_z;
    }
    if(mode & 0x8) {
       Vector3 quat(quat_bytes_minus1);
    }
}
```

7.3.2 Semantics

`abs_flag`: If 1, absolute position and orientation is specified. If 0, the specified values are added relative to the previously coded position and orientation.

`mode`: Signalling mode; Valid values are:

- `[1, 7]`: Only the position is signalled.
- `8`: Only the orientation is signalled.
- `[9, 15]`: Both, orientation and position are signalled.

`pos_bytes_minus1`: Plus 1 indicates the number of bytes to be read for `pos_x`, `pos_y` and `pos_z`. Valid values are in the range from `[0, 3]`.

`pos_unit`: Unit of `pos_x`, `pos_y` and `pos_z`. Valid values are in the range from `[0, 2]`, where

0: µm

1: mm

2: m

`quat_bytes_minus1`: Plus 1 indicates the number of bytes to be read for `quat.x`, `quat.y`, `quat.z`. Valid values are in the range from `[0, 1]`.

`quat_den_bits_minus1`: Specifies the denominator of `quat.x`, `quat.y` and `quat.z`. Valid values for `quat_den_bits_minus1` are in the range from `[0, 13]`. The denominator is computed as follows:

$$\texttt{denominator} = 2^{\textbf{quat\_den\_bits\_minus1}+1}$$

`pos_x`: Specifies the x-coordinate of the location of the camera in units specified by `pos_unit`. When not present, its value shall be inferred to be 0 if `abs_flag` is 1.

`pos_y`: Specifies the y-coordinate of the location of the camera in units specified by `pos_unit`. When not present, its value shall be inferred to be 0 if `abs_flag` is 1.

`pos_z`: Specifies the z-coordinate of the location of the camera in units specified by `pos_unit`. When not present, its value shall be inferred to be 0 if `abs_flag` is 1.

`quat.x`: Specifies the x component, `qX`, for the rotation of the camera using the quaternion representation. The range of `quat_x` shall be in the range of $-2^{\texttt{quat\_den\_bits\_minus1}+1}$ to $2^{\texttt{quat\_den\_bits\_minus1}+1}$, inclusive. When not present, its value shall be inferred to be 0 if `abs_flag` is set to 1.

`quat.y`: Specifies the y component, `qY`, for the rotation of the camera using the quaternion representation. The range of `quat_y` shall be in the range of $-2^{\texttt{quat\_den\_bits\_minus1}+1}$ to $2^{\texttt{quat\_den\_bits\_minus1}+1}$, inclusive. When not present, its value shall be inferred to be 0 if `abs_flag` is set to 1.

`quat.z`: Specifies the z component, `qZ`, for the rotation of the camera using the quaternion representation. The range of `quat_z` shall be in the range of $-2^{\texttt{quat\_den\_bits\_minus1}+1}$ to $2^{\texttt{quat\_den\_bits\_minus1}+1}$, inclusive. When not present, its value shall be inferred to be 0 if `abs_flag` is set to 1.

The values of the quaternion representation are computed as follows:

```
qX = quat.x / denominator

qY = quat.y / denominator

qZ = quat.z / denominator
```

It is a requirement of bitstream conformance that:

```
qX2 + qY2 +qZ2 <= 1
```

The fourth component of the quaternion representation, `qW`, is computed as follows:

$$qW = Sqrt(1 - (qX^2 + qY^2 + qZ^2))$$

The point `(w, x, y, z)` represents a rotation around the axis directed by the vector `(x, y, z)` by an angle `2*cos ^{-1}(w)=2*sin ^{-1}(sqrt(x^{2}+y^{2}+z^{2}))`.

NOTE As aligned ISO/IEC FDIS 23090-5, qW is always positive. If a negative qW is desired, one can signal all three syntax elements, cam_quat_x, cam_quat_y, and cam_quat_z with an opposite sign, which is equivalent.

7.4 Intrinsic Camera Information

Intrinsic camera information is defined using the following data structure.

7.4.1 Syntax

```
aligned(8) class IntCameraInfo (unsigned char precision_bytes_minus1) {
   unsigned int(10) camera_id;
   bit(3) reserved = 0;
   unsigned int(3) camera_type;
   if (camera_type == 0) {
      signed int((precision_bytes_minus1+1)*8) erp_horizontal_fov;
      signed int((precision_bytes_minus1+1)*8) erp_vertical_fov;
   }
   if (camera_type == 1) {
      signed int((precision_bytes_minus1+1)*8) perspective_horizontal_fov;
      unsigned int(8)[4] perspective_aspect_ratio;
   }
   if (camera_type == 2) {
      unsigned int(8)[4] ortho_aspect_ratio;
      unsigned int(8)[4] ortho_horizontal_size;
   }
   unsigned int(8)[4] clipping_near_plane;
   unsigned int(8)[4] clipping_far_plane;
}
```

7.4.2 Semantics

`camera_id` is an identifier number that is used to identify a given viewport camera parameters.

`camera_type` indicates the projection method of the viewport camera. The value 0 specifies ERP projection. The value 1 specifies a perspective projection. The value 2 specifies an orthographic projection. Values in the range 3 to 255 are reserved for future use by ISO/IEC.

`precision_bytes_minus1`: Plus 1 indicates the number of bytes to be read for `erp_horizontal_fov`, `erp_vertical_fov` and `perspective_horizontal_fov`. Valid values are in the range from `[0, 3]`.

`erp_horizontal_fov` specifies the longitude range for an ERP projection corresponding to the horizontal size of the viewport region, in units of radians. The value shall be in the range 0 to 2π.

`erp_vertical_fov` specifies the latitude range for an ERP projection corresponding to the vertical size of the viewport region, in units of radians. The value shall be in the range 0 to π.

`perspective_horizontal_fov` specifies the horizontal field of view for perspective projection in radians. The value of shall be in the range of 0 and π.

`perspective_aspect_ratio` specifies the relative aspect ratio of viewport for perspective projection (horizontal/vertical). The value shall be expressed in 32-bit binary floating-point format with the 4 bytes in big-endian order and with the parsing process as specified in IEEE 754.

`ortho_aspect_ratio` specifies the relative aspect ratio of viewport for orthogonal projection (horizontal/vertical). The value shall be expressed in 32-bit binary floating-point format with the 4 bytes in big-endian order and with the parsing process as specified in IEEE 754.

`ortho_horizontal_size` specifies the horizontal size of the orthogonal in metres. The value shall be expressed in 32-bit binary floating-point format with the 4 bytes in big-endian order and with the parsing process as specified in IEEE 754.

`clipping_near_plane` and `clipping_far_plane` indicate the near and far depths (or distances) based on the near and far clipping planes of the viewport in metres. The values shall be expressed in 32-bit binary floating-point format with the 4 bytes in big-endian order and with the parsing process as specified in IEEE 754.

7.5 Viewing Spaces

A cuboid viewing space is defined using the following data structure.

7.5.1 Syntax

```
aligned(8) class ViewingSpace(unsigned char precision_bytes_minus1) {
   Vector3 anchor((precision_bytes_minus1+1)*8);
   Vector3 dimensions((precision_bytes_minus1+1)*8);
}
```

7.5.2 Semantics

`precision_bytes_minus1`: Plus 1 indicates the number of bytes to be read for `anchor`, and `dimensions`. Valid values are in the range from `[0, 3]`.

`anchor.x`, `anchor.y`, and `anchor.z` indicate the x, y, z position values of the anchor point of the viewing space, respectively, relative to the origin `(0,0,0)`.

`dimensions.x`, `dimensions.y`, and `dimensions.z` indicate the dimensions (or ranges) in the Cartesian coordinates along the x, y, and z axes, respectively, from the anchor `(anchor.x, anchor.y, anchor.z)`.

7.6 Cuboid Regions

A cuboid region is defined using the following data structure.

7.6.1 Syntax

```
aligned(8) class CuboidRegion (
      unsigned int(1) anchor_included,
      unsigned int(1) scale_included,
      unsigned char precision_bytes_minus1) {
   unsigned int(16) id;
   if (anchor_included) { // when anchor is not 0,0,0
      Vector3 anchor((precision_bytes_minus1+1)*8);
   }
   if (scale_included) { // when scale is not (1,1,1)
      Vector3 scale((precision_bytes_minus1+1)*8);
   }
   Vector3 dimensions((precision_bytes_minus1+1)*8);
}
```

7.6.2 Semantics

`id`: is an identifier of the region.

`precision_bytes_minus1`: Plus 1 indicates the number of bytes to be read for `anchor`, `scale` and `dimensions`. Valid values are in the range from `[0, 3]`.

`anchor.x`, `anchor.y`, and `anchor.z` indicate the x, y, z position values of the anchor point of the viewing space, respectively, relative to the origin `(0,0,0)`. The default is `(0,0,0)`.

`scale.x`, `scale.y`, and `scale.z` indicate the scaling extension in the Cartesian coordinates along the x, y, and z axes, respectively, relative to the origin `(0,0,0)`. The default is `(1,1,1)`.

`dimensions.x`, `dimensions.y`, and `dimensions.z` indicate the dimensions (or ranges) in the Cartesian coordinates along the x, y, and z axes, respectively, from to the anchor `(anchor.x, anchor.y, anchor.z)`.

7.7 3D Region Set

7.7.1 Definition

A 3D region set is a structure that defines a part of a volumetric media as a set of one or more regions. Each region of the set is specified as a geometry that defines the shape, position and size of this region inside a reference space that is mapped to the volumetric media with which the 3D region set is associated.

The geometry of a region described by the 3D region set can be represented either by:

— a point

— a polyline

— a plane

— a rectangular cuboid

— a value for an attribute of the volumetric media.

The part of the volumetric media that is contained in a region depends on the geometry of the region as follows:

— when the geometry of a region is represented by a point, the region contains the location of this point;

— when the geometry of a region is represented by a polyline, the region contains the lines that are part of the polyline;

— when the geometry of a region is represented by a plane, the region contains the surface of the plane included in the bounding box of the volumetric media;

— when the geometry of a region is represented by a rectangular cuboid, the region contains the volume inside this rectangular cuboid, including its faces;

— when the geometry of a region is represented by a value for an attribute of the volumetric media, the region contains all the points from the volumetric media whose attribute's value is the same as the value specified for the region.

A region may be empty if it falls entirely outside the bounding box of the volumetric media with which it is associated. An empty region should be ignored.

7.7.2 Syntax

```
aligned (8) class 3DRegionSet(version, flags) {
   unsigned int precision = flags & 3;
   unsigned int(8) region_count;
   for (r=0; r < region_count; r++) {
   unsigned int(8) geometry_type;
   if (geometry_type == 0) {
      // Use a 3D Point from Part-7.
      Vector3 anchor(precision);
   }
   if (geometry_type == 1) {
      // polyline
      unsigned int(field_size) point_count;
      for (i=0; i < point_count; i++) {
      // Use a 3D Point from Part-7.
```

```
        Vector3 point(precision);
        }
    }
    if (geometry_type == 2) {
        // plane: to be defined in Part-7.
        Vector3 anchor(precision);
        Vector3 normal(precision);
    }
    if (geometry_type == 3) {
        // rectangular cuboid
        // Defined in Part-7.
        CuboidRegion cuboid(1, 0, precision);
        QuaternionRotation rotation();
    }
    // … possibly other 3D shapes
    if (geometry_type == 5) {
        // region defined by an attribute
        // Bounding box of the region. Could be optional.
        CuboidRegion cuboid(1, 0, precision)
        unsigned int(8) region_identifier_value;
    }
}
```

7.7.3 Semantics

`version` shall be equal to 0.

`flags`: the 2 least-significant bits define the precision used by the different geometry attributes of the 3D region set. The values of flags greater than 3 are reserved.

`region_count` is the number of regions contained in this 3D region set.

`geometry_type` specifies the type of the geometry of a region. The following values for `geometry_type` are defined:

— 0: the region is described as a point

— 1: the region is described as a polyline

— 2: the region is described as a plane

— 3: the region is described as a rectangular cuboid

— 5: the region is described using an attribute of the volumetric media

— other values are reserved.

`anchor` specifies the position of the point composing a region when its geometry is a point. It specifies the position of a point contained in the plane composing a region when its geometry is a plane.

`point` specifies the position of a point belonging to the polyline composing a region when its geometry is a polyline.

`normal` specifies the normal vector of the plane composing a region when its geometry is a plane.

`cuboid` specifies the rectangular cuboid composing a region when its geometry is a rectangular cuboid. It specifies a bounding box for a region when its geometry is described using an attribute of the volumetric media.

`rotation` specifies the rotation applied to the cuboid composing a region when its geometry is a rectangular cuboid.

`region_identifier_value` specifies the value of the attribute defining a region when its geometry is described using an attribute of the volumetric media.

7.8 Volumetric region item

7.8.1 Definition

An item with an item_type value of 'vran' is a volumetric region item that defines one or more regions of an item containing volumetric media.

A volumetric region item allows associating a same set of item properties or other items or both with each individual region it defines inside a volumetric media stored inside an item. Item properties should only be associated with a volumetric region item when the property value for the region differs from the matching (explicit or implied) property value for the whole volumetric media.

The volumetric region item is associated with the item inside which the regions are defined using an item reference of type 'cdsc' from the volumetric region item to the item containing the volumetric media.

The regions described by the volumetric region item defined in `VolumetricRegionItem` are specified in the data of the volumetric region item.

Annex A describes how a volumetric region item with item_type value of 'vran' shall be used to associate annotations with spatial areas of non-timed visual volumetric data encoded and stored in a V3C item.

Annex B describes how a volumetric region item with item_type value of 'vran' shall be used to associate annotations with spatial areas of non-timed G-PCC data encoded and stored in a G-PCC item.

7.8.2 Syntax

```
aligned (8) class VolumetricRegionItem {
   unsigned int(8) version = 0;
   unsigned int(8) flags;
   3DRegionSet regions(version, flags);
}
```

7.8.3 Semantics

`version` shall be equal to 0. Readers shall not process a `VolumetricRegionItem` with an unrecognized version number.

`flags` is used to control the contents of the `3DRegionSet` structure used for defining the content of the volumetric region item.

`regions` is the structure defining the regions contained in the volumetric region item using the `3DRegionSet` defined in ISO/IEC 23090-7.

Annex A and Annex B

Add the following Annex A and Annex B after the newly added Clause 7.