**ISO/IEC 23008-12:2022/AMD 1:2022(E)**

ISO/IEC JTC1/SC 29

Secretariat: JISC

**Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 12: Image File Format — Amendment 1: Support for progressive rendering signalling and other improvements**

draftDIS stage

**Copyright notice**

This ISO document is a working draft or committee draft and is copyright-protected by ISO. While the reproduction of working drafts or committee drafts in any form for use by participants in the ISO standards development process is permitted without prior permission from ISO, neither this document nor any extract from it may be reproduced, stored or transmitted in any form for any other purpose without prior written permission from ISO.

Requests for permission to reproduce this document for the purpose of selling it should be addressed as shown below or to ISO's member body in the country of the requester:

ISO copyright office

Case postale 56 • CH-1211 Geneva 20

Tel. + 41 22 749 01 11

Fax + 41 22 749 09 47

E-mail copyright@iso.org

Web www.iso.org

Reproduction for sales purposes may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

Contents

[Foreword iv](#_Toc120694394)

[3.1.AA region 1](#_Toc120694395)

[3.1.BB region annotation 1](#_Toc120694396)

[3.1.CC region track 1](#_Toc120694397)

[3.1.DD progressive rendering 1](#_Toc120694398)

[3.1.EE progressive refinement 1](#_Toc120694399)

[3.1.FF progressive decoding 1](#_Toc120694400)

[6.5.37 Progressive derived image item information 4](#_Toc120694401)

[6.5.38 Single stream 5](#_Toc120694402)

[6.5.39 Camera extrinsic matrix 6](#_Toc120694403)

[6.5.40 Camera intrinsic matrix 9](#_Toc120694404)

[6.8.10 Progressive rendering entity group 10](#_Toc120694405)

[11. Region and region annotation 11](#_Toc120694406)

[11.1. Overview 11](#_Toc120694407)

[11.2. Common definitions for image sequence or video tracks and for image items 11](#_Toc120694408)

[11.2.1. Region geometry structure 11](#_Toc120694409)

[11.2.2. Mask item 16](#_Toc120694410)

[11.3. Regions and region annotations for an image item 17](#_Toc120694411)

[11.3.1. General 17](#_Toc120694412)

[11.3.2. Region item 18](#_Toc120694413)

[11.3.3. Derived region item 19](#_Toc120694414)

[11.4. Regions and region annotations for an image sequence or a video track 20](#_Toc120694415)

[11.4.1. General 20](#_Toc120694416)

[11.4.2. Region track 20](#_Toc120694417)

[11.4.3. Sample groups for region track 22](#_Toc120694418)

[H.3.2 JPEG configuration box 25](#_Toc120694420)

[Annex K (informative) Examples of progressive decoding, rendering and refinement 27](#_Toc120694421)

[K.1 Overview 27](#_Toc120694422)

[K.2 Remarks on creating a file suitable for progressive refinement 28](#_Toc120694423)

[K.3 Progressive rendering and refinement with independently coded image items 28](#_Toc120694424)

[K.4 Region-wise progressive rendering and refinement with independently coded image items 28](#_Toc120694425)

[K.5 Overlay derived image item represented by a single bitstream with temporal inter prediction 30](#_Toc120694426)

[K.6 Multi-layer images with the progressive rendering entity group 31](#_Toc120694427)

[K.7 Player operation for progressive rendering or refinement for HEIF files 32](#_Toc120694428)

[K.7.1 Explicit signalling for progressive rendering or refinement 33](#_Toc120694429)

[K.7.2 Implicit signalling for progressive rendering or refinement 33](#_Toc120694430)

[K.7.3 Progressive rendering or refinement steps 33](#_Toc120694431)

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 documents should be noted.  This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see [www.iso.org/directives](http://www.iso.org/directives) or [www.iec.ch/members\_experts/refdocs](http://www.iec.ch/members_experts/refdocs)).

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

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

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

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

A list of all parts in the ISO/IEC 23008 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](https://www.iso.org/members.html) and [www.iec.ch/national-committees](http://www.iec.ch/national-committees).

Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 12: Image File Format — Amendment 1: Support for progressive rendering signalling and other improvements

*3*

Rename the clause 3 as “Terms, definitions, abbreviated terms and symbols”

Add the following definitions in subclause 3.1 (“*Terms and definitions*”) and renumbered all definitions and cross-references accordingly

3.1.AA   
region

area represented by a shape, position and size encompassing a part of an *image* (3.1.16)

3.1.BB   
region annotation

metadata or data representing an annotation associated with a *region* (3.1.AA)

3.1.CC   
region track

track whose samples define a *region* (3.1.AA) within samples of another track with which the track is associated via track reference

3.1.DD   
progressive rendering

displaying an *image item* (3.1.18) or a sample in successive steps where each step improves the perceived image quality over that of the previous step and is superimposed over the image (3.1.16) of the previous step in the same displaying window

Note 1 to entry: A progressive rendering step can improve the perceived image quality over the complete image as a whole, or region by region resulting in a region-wise progressive rendering

3.1.EE   
progressive refinement

*progressive rendering* (3.1.DD) of an *image item* (3.1.18) or sample in a file while downloading the file

3.1.FF   
progressive decoding

decoding a bitstream with a single decoder instance in successive steps where each step improves the perceived image quality over that of the previous step

Replace the following in subclause 3.1.9 (“*derived region item*”)

*item* (3.1.27) whose data is a representation of the shape, position and size of a region within an *image item* (3.1.18), with which it is associated via item reference, as an *operation* (3.1.33) on other *region items* (3.1.39)

with

*item* (3.1.27) whose data defined a *region* (3.1.AA) within an *image item* (3.1.18), with which the item is associated via item reference, as an *operation* (3.1.33) on other *region items* (3.1.39)

Replace the following in subclause 3.1.39 (“*region item*”)

*item* (3.1.27) whose data defines the shape, position and size of a region within an *image item* (3.1.18) with which it is associated via item reference

with

*item* (3.1.27) whose data defines a *region* (3.1.AA) within an *image item* (3.1.18) with which the item is associated via item reference

Add the following new subclause after subclause 3.2 (*“Abbreviated terms”*):

* 1. **Symbols**
     1. **Mathematical functions**

abs( x ) (1)

sqrt( x ) the square root of x

*6*

Remove the entire subclauses 6.10 (“*Region items and region annotations*”) and 6.11 (“*Derived region items*”) (Editor’s note: contents of those subclauses are integrated into a new subclause 11 on “*Regions and region annotations*”)

*6.5*

Replace the following in subclause 6.5.1 (“*General*”)

Descriptive properties are non-essential, unless stated otherwise in their specification.

with

Descriptive properties, regardless of being specified in this document or in any derived specification, should be marked as non-essential, unless stated otherwise in their definition.

Replace the following in subclause 6.5.1 (“*General*”)

Properties may be associated with an entity group, but only when explicitly stated in their specification.

with

Properties may be associated with an entity group, but only when explicitly stated in their definition.

Replace the content of subclause 6.5.5.1 (“*Definition*” for “*Colour information”*) with the following

|  |  |
| --- | --- |
| Box type: | 'colr' |
| Property type: | Descriptive item property |
| Container: | ItemPropertyContainerBox |
| Mandatory (per item): | No |
| Quantity (per item): | Either at most one, or two with restriction described below |
|  |  |

The definition of Colour Information provided in ISO/IEC 14496-12 applies.

In addition, following definitions specific to the use of colour information in image items also applies:

* When two ColourInformationBoxes are associated with an image item, one shall have a colour\_type value of 'rICC' or 'prof' (providing either restricted or unrestricted ICC profiles respectively) and the other one shall have a colour\_type value of 'nclx' with colour\_primaries equal to 2 and transfer\_characteristics equal to 2 (2 indicating "unspecified", since these data are supplied by the ICC profile instead).
* When generating an image item from the content of a visual track, the order of ColourInformationBoxes in the VisualSampleEntry should be preserved in the ItemPropertyAssociationBox(es). Similarly, when creating a visual track from an image item, the order of boxes should be preserved.

While in a visual track, the order of boxes may be important per ISO/IEC 14496-12, in this specification, the order is not relevant. Colour information with different values of colour\_type are intended for different purposes. Colour information with a value of colour\_type set to 'nclx' is intended to be used for some processing such as colour conversion or image derivation, while colour information carrying an ICC profile is intended to be used for processes such as display matching.

Add the following new subclauses after subclause 6.5.36:

### Progressive derived image item information

#### Definition

|  |  |
| --- | --- |
| Box type: | 'prdi' |
| Property type: | Descriptive item property |
| Container: | ItemPropertyContainerBox |
| Mandatory (per item): | No |
| Quantity (per item): | Zero or one for a derived image item |
|  |  |

The ProgressiveDerivedImageItemInformationProperty describes progressive rendering steps associated with a derived image item.

NOTE The ProgressiveDerivedImageItemInformationProperty is intended to be used with derived image items using several input images.

Each progressive rendering step specifies the number of input image items to use for the reconstruction of the derived image item and is described as a difference from the previous step.

The sum of item\_count(s) over all progressive rendering steps step\_count shall be greater than or equal to the number of input image items listed in the 'dimg' item reference of the derived image item. For instance, this sum may be greater when the input image items have alternative(s) (e.g. one or more associated thumbnails or when an input image item belongs to an 'altr' entity group). In such case, the list of input image items and their alternatives should be ordered to allow a progressive rendering from the lowest quality to the highest quality of the derived image item.

The flags field is used to signal the intended progressive rendering behaviour. The following flags values are defined:

0x000001 item\_reference\_order; when set, it indicates that the input image items to use for a progressive rendering step follow the declaration order in the 'dimg' item reference of the derived image item. When not set, the input image items to use for a progressive rendering step follow the order of appearance of the input items in the file.

0x000002 one\_item\_per\_step; when set, it indicates that each progressive rendering step consumes only 1 input image item. When not set, the number of items consumed by the *i*th progressive rendering step is given by the *i*th item\_count.

#### Syntax

aligned(8) class ProgressiveDerivedImageItemInformationProperty  
extends ItemFullProperty('prdi', version = 0, flags){  
 unsigned int(16) step\_count;  
 if ((flags & one\_item\_per\_step) == 0) {  
 for (i=0; i < step\_count; i++)  
 unsigned int(16) item\_count;  
 }  
}

#### Semantics

step\_count is the number of progressive steps for the associated derived image item.

NOTE When step\_count is set to 1, it means that progressive rendering is not desired.

item\_count is the number of input image items added by the *i*th progressive step. When not present, item\_count is inferred to be equal to 1.

### Single stream

#### Definition

|  |  |
| --- | --- |
| Box type: | 'sstr' |
| Property type: | Descriptive item property |
| Container: | ItemPropertyContainerBox |
| Mandatory (per item): | No |
| Quantity (per item): | Zero or one for a derived image item |
|  |  |

The SingleStreamProperty associated with a derived image item indicates that the item data of the input image items collectively form a single bitstream that is conformant to the coding format of the input image items and is decodable with a single decoder. The single bitstream is referred to as the derived bitstream in this subclause.

When a derived image item has an associated SingleStreamProperty, the derived bitstream resulting by processing the input image items as specified in the next paragraph shall conform to the item type and decoder configuration item property of the last input image item to the derived image item.

The order in which input image items are listed in the 'dimg' item reference of a derived image item is the order in which the item data of the input image items are to be concatenated to obtain the derived bitstream. When an input image is a predictively coded image item, each one of its reference image items is included in the order of 'pred' item references into the derived bitstream unless it has already been included in the derived bitstream, followed by including the item data of the predictively coded image item into the derived bitstream.

NOTE 1 The SingleStreamProperty is intended to be used with derived image items using several input images, for example, in progressive rendering. Clause K.5 describes an example of the usage of SingleStreamProperty with an overlay derived image item for progressive rendering. In this example, each decoded image in the derived bitstream corresponds to a rendering step in progressive rendering. When the derived bitstream conforms to VVC, the spatial placement of each decoded image can be achieved through VVC scaling windows as illustrated in Clause K.5.

NOTE 2 When a file writer assigns the SingleStreamProperty to a derived image item, the file writer needs to provision the item type and the decoder configuration record of the last input image item to be applicable to both the last input image item and the derived bitstream. Moreover, when parameter sets are included in the decoder configuration record of the last input image item, the file writer needs to provision the parameter sets to be applicable to both the last input image item and the derived bitstream. When a file writer assigns the SingleStreamProperty to a derived image item and sequence-level parameter sets are included in item data of the input image items, the file writer needs to verify that all the sequence-level parameter sets in the input image items that end up being in the same coded video sequence in the derived bitstream are identical.

#### Syntax

aligned(8) class SingleStreamProperty  
extends ItemProperty ('sstr'){  
}

### Camera extrinsic matrix

#### Definition

|  |  |
| --- | --- |
| Box type: | 'cmex' |
| Property type: | Descriptive item property |
| Container: | ItemPropertyContainerBox |
| Mandatory (per item): | No |
| Quantity (per item): | Zero or more |
|  |  |

The CameraExtrinsicMatrixProperty describes the spatial setup of camera(s). It specifies a position, in the cartesian representation, and an orientation, in the quaternion representation, of an orthogonal right-handed camera coordinate system within an orthogonal right-handed cartesian 3D world coordinate system.

NOTE 1 When needed, the information on the cameras GPS position can be provided using EXIF as defined in Annex A.

The camera coordinate system is relative to the camera’s origin and orientation where the z-axis of the camera coordinate system faces outward from the camera lens.

The world coordinate system refers to the scene in which all the camera systems are positioned and can be identified, when several world coordinate systems occur in one file. The origin may be chosen to be at any point since the main intent of the camera extrinsics is to describe the relative positioning of the camera coordinate systems. It is recommended that the origin is placed either at one of the cameras in a multi-camera setup, or in the geometric centre.

The flags field is used to allow only specifying a sub-set of the extrinsic properties for common use-cases. The following flags are defined:

0x000001 pos-x-present; indicates that the position is signalled along the x axis.

0x000002 pos-y-present; indicates that the position is signalled along the y axis.

0x000004 pos-z-present; indicates that the position is signalled along the z axis.

0x000008 orientation-present; indicates that orientation is signalled.

0x000010 rot-large-field-size; indicates that orientation elements are 32-bit integers (otherwise 16-bit integers).

0x000020 id-present; indicates that the world coordinate system id is signalled.

Other flags values are reserved.

Examples of use-cases are:

* Simple stereo pair (typically only requires a baseline):
  + Use flags value pos-x-present.
* Multiple cameras facing the same direction (typically requires a position in 2 or 3 axes):
  + Use flags value including at least one of pos-x-present, pos-y-present or pos-z-present (depending on which axes are non-zero).
* Panorama image collections (typically only requires orientation):
  + Use flags value orientation-present.
* 3D mapping image collections (typically requires both 3D position and orientation):
  + Use flags values pos-x-present, pos-y-present, pos-z-present and orientation-present.

CameraExtrinsicMatrixProperty instances with the same id value indicate that the associated image items were captured in the same world coordinate system.

When version equals 0, the following applies:

* The variable *orientationPrecision* is set equal to ((flags & rot-large-field-size)?16:0).
* The values of the quaternion representation are computed as follows:

*qX* = quat\_x / 214 + *orientationPrecision*

*qY* = quat\_y / 214 + *orientationPrecision*

*qZ* = quat\_z / 214 + *orientationPrecision*

NOTE 2 Formulas above use a floating-point division, not an integer division.

* It is a requirement of bitstream conformance that:

*qX*2 + *qY*2 + *qZ*2 <= 1

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

*qW* = abs( sqrt( 1 – ( *qX*2 + *qY*2 + *qZ*2 ) )

NOTE 3 In the context of this specification *qW* is always positive. If a negative *qW* is desired, one can signal all three syntax elements, quat\_x, quat\_y and quat\_z, with an opposite sign, which is equivalent.

* The camera extrinsic matrix signalled with the CameraExtrinsicMatrix describes the transformation of the camera coordinate system relative to the world coordinate system with the rotation matrix based on the unit quaternion as follows:

and the camera position signaled as:

where:

*pos\_x* = pos\_x

*pos\_y* = pos\_y

*pos\_z =* pos\_z

* To describe the extrinsic matrix with the rotation matrix and the position vector , which describes the transformation of the world coordinate system relative to the camera coordinate system, the following calculation can be used:

where is the transpose of the matrix .

When version equals 1, the orientation of the camera coordinate system relative to the world coordinate system is provided by ViewpointGlobalCoordinateSysRotationStruct data structure where the camera coordinate system is represented by the global coordinate system of the viewpoint defined in ISO/IEC 23090-7 and where the world coordinate system corresponds to the common reference coordinate system defined in ISO/IEC 23090-7.

#### Syntax

aligned(8) class CameraExtrinsicMatrixProperty  
extends ItemFullProperty('cmex', version, flags) {  
 if (flags & pos-x-present) {  
 signed int(32) pos\_x;  
 }  
 if (flags & pos-y-present) {  
 signed int(32) pos\_y;  
 }  
 if (flags & pos-z-present) {  
 signed int(32) pos\_z;  
 }  
 if (flags & orientation-present) {  
 if (version == 0) {  
 signed int((flags & rot-large-field-size )?32:16) quat\_x;  
 signed int((flags & rot-large-field-size )?32:16) quat\_y;  
 signed int((flags & rot-large-field-size )?32:16) quat\_z;  
 } else if (version == 1) {  
 ViewpointGlobalCoordinateSysRotationStruct rot;  
 }  
 }  
 if (flags & id-present) {  
 unsigned int(32) id;  
 }

}

#### Semantics

pos\_x specifies the x-coordinate of the location of the camera in µm. When not present, its value shall be inferred to be 0.

pos\_y specifies the y-coordinate of the location of the camera in µm. When not present, its value shall be inferred to be 0.

pos\_z specifies the z-coordinate of the location of the camera in µm. When not present, its value shall be inferred to be 0.

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 -2N to 2N, inclusive (where N is 14 + *orientationPrecision*). When not present, its value shall be inferred to be 0.

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 -2N to 2N, inclusive (where N is 14 + *orientationPrecision*). When not present, its value shall be inferred to be 0.

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 -2N to 2N, inclusive (where N is 14 + *orientationPrecision*). When not present, its value shall be inferred to be 0.

rot specifies the rotation in yaw, pitch, and roll rotation angles of the camera as defined in ISO/IEC 23090-7. When not present, its values in yaw, pitch, and roll rotation angles shall be inferred to be 0.

id specifies the coordinate system identifier. When not present, its value shall be inferred to be 0. If more than one CameraExtrinsicMatrixProperty is associated with the same item, the id shall be present.

### Camera intrinsic matrix

#### Definition

|  |  |
| --- | --- |
| Box type: | 'cmin' |
| Property type: | Descriptive item property |
| Container: | ItemPropertyContainerBox |
| Mandatory (per item): | No |
| Quantity (per item): | Zero or one |
|  |  |

The CameraIntrinsicMatrixProperty describes the characteristics of the camera that captured the associated image item.

The intrinsic matrix for a pinhole camera is specified as follows:

|  |  |  |
| --- | --- | --- |
| *fx* | *s* | *cx* |
| *0* | *fy* | *cy* |
| *0* | *0* | *1* |

where:

*fx*: horizontal focal length  
*fy*: vertical focal length  
*s*: skew factor  
*cx*: principal point x  
*cy*: principal point y

NOTE 1 For most cameras, pixels are square and there is no skew. This corresponds to *s* being zero and *fx* being equal to *fy*.

The flags field is used to define the values of denominator and skew denominator.

The variable *denominator* is set equal to (1 << *denominatorShiftOperand*) where *denominatorShiftOperand* is equal to ((flags & 0x001F00) >> 8).

The variable *skewDenominator* is set equal to (1 << *skewDenominatorShiftOperand)* where *skewDenominatorShiftOperand* is equal to((flags & 0x1F0000) >> 16).

The values of the above intrinsic matrix can be calculated as follows:

*fx* = focal\_length\_x × image\_width / *denominator*

*fy* = focal\_length\_y × image\_height / *denominator*

*cx* = principal\_point\_x × image\_width / *denominator*

*cy* = principal\_point\_y × image\_height / *denominator*

*s* = skew\_factor / *skewDenominator*

where

image\_width and image\_height come from the ImageSpatialExtentsProperty associated with the image item.

NOTE 2 By specifying the focal lengths and principal point as normalized by image dimensions, this allows for the intrinsic matrix to be scale invariant. For a camera system without skew, this means that the same intrinsic matrix can be used even if the sensor uses pixel binning to output images with varying number of pixels.

NOTE 3 Formulas above use a floating-point division, not an integer division.

#### Syntax

aligned(8) class CameraIntrinsicMatrixProperty  
extends ItemFullProperty('cmin', version = 0, flags) {  
 signed int(32) focal\_length\_x;  
 signed int(32) principal\_point\_x;  
 signed int(32) principal\_point\_y;  
 if (flags & 1) {  
 signed int(32) focal\_length\_y;  
 signed int(32) skew\_factor;  
 }  
}

#### Semantics

(flags & 1) equal to 0 indicates that simplified intrinsics (no skew, square pixels) are used. (flags & 1) equal to 1 indicates that full intrinsics are used.

focal\_length\_x specifies the horizontal focal length of the camera in image widths.

focal\_length\_y specifies the vertical focal length of the camera in image heights. When not present, the value of *fy* is inferred to be equal to *fx*.

principal\_point\_x specifies the principal point x-coordinate in image widths.

principal\_point\_y specifies the principal point y-coordinate in image heights.

skew\_factor specifies the camera system skew factor. When not present its value shall be implied to be 0.

*6.8*

Add the following new subclause after subclause 6.8.9:

### Progressive rendering entity group

The progressive rendering entity group (with a grouping\_type 'prgr') signals a set of image items that can be used for a progressive rendering of one of these image items.

The semantics of the 'prgr' entity group are that the image items included in a 'prgr' entity group are listed in increasing quality order from the lowest quality to the highest quality. All the image items inside a 'prgr' entity group shall correspond to similar images albeit with different quality levels. In this way, a first image item occurring earlier in the list than a second image item can be used as a temporary replacement of the second image item for a progressive rendering of this second image item.

The data corresponding to the image items included in a 'prgr' entity group shall be stored in the same order as the one used for the image items inside the 'prgr' entity group, such that a renderer progressively obtaining a file can perform a progressive refinement as item data becomes available.

A 'prgr' entity group shall only contain image items, not tracks.

[Editor’s note: In response to FR12-040 ballot comment on CDAM1 wondering whether we should allow tracks in ‘prgr’ since allowed in ‘altr’ entity group, we need to study what progressive rendering of tracks means.]

Image items of the same 'prgr' entity group shall be members of the same 'altr' entity group.

NOTE This requirement guarantees that legacy players without capability of processing 'prgr' entity groups treat the image items as alternatives to be displayed.

*10*

Add the following new subclause after subclause 10:

# Region and region annotation

## Overview

This clause specifies tools to associate annotations, e.g. metadata or images with one or more regions of an image.

Subclause 11.2 specifies common structures to describe the geometry (position, size, shape) of a region as usual geometrical shapes or as a mask.

Subclause 11.3 specifies how to describe regions of an image carried in an image item and how to associate region annotations to each region.

Subclause 11.4 specifies how to describe regions of an image carried in an image sequence or video track and how to associate region annotations to each region.

## Common definitions for image sequence or video tracks and for image items

### Region geometry structure

#### Definition

RegionGeometryStruct() specifies the geometries of one or more regions inside images depending on the container of the images:

* For an image carried in an image item, the container of this structure is a region item, as defined in 11.3.2, and this structure defines one or more regions in the image carried by the image item (also denoted source image item in the following).
* For images carried in an image sequence or video, the container of this structure is a sample of a region track, as defined in 11.4.2, and this structure defines one or more regions inside images carried in samples of the image sequence or video track (also denoted source track in the following).

These geometries define the shape, position and size of the regions inside a reference space that is mapped to the image.

The geometry of a region inside the image is obtained after applying the implicit resampling caused by the difference between the size of the reference space, as defined in 11.3.2 and 11.4.2, and the size of the image. The resampling shall occur after applying both of the following:

* Applying on the reference space and the geometry of region all transformations associated with the region item or the region track, and
* Applying on the image all transformations associated with the source image item or the source track.

NOTE 1 The implicit resampling can modify the declared aspect ratio of the reference space.

NOTE 2 Transformations on region track or source track include CleanApertureBox (when present), implicit rescaling to the width and height of the TrackHeaderBox, and the matrix of the TrackHeaderBox as specified in ISO/IEC 14496-12.

The geometry of a region can be represented either by:

* a point,
* a polyline,
* a rectangle,
* an ellipse,
* a polygon,
* a mask stored in the sample of another track or in an image item, or,
* a mask defined in the data of the structure.

Line segments for both polygons and polylines shall not cross (including the implicit closing line for polygons).

The pixels of the image that are part of a region depend on the geometry of the region as follows:

* When the geometry of a region is represented by a point, the pixel located at this point, if it exists, is part of the region.
* When the geometry of a region is represented by a rectangle, an ellipse, or a polygon, the pixels that are inside (including the boundary) of the rectangle, ellipse, or polygon are part of the region.
* When the geometry of a region is represented as a mask defined in the data of the sample of the region track (i.e., when geometry\_type equals 5), the pixels of the sample of the associated source track corresponding to pixels with the value 1 in the mask image are part of the region.
* When the geometry of a region is represented by a polyline, the pixels are part of the region when they would be coloured or partly coloured (e.g. when using anti-aliasing) if a one-pixel-wide line were drawn along the polyline.
* When the geometry of a region is represented by a mask stored in another entity(i.e., when geometry\_type equals 4), the entity containing the mask is identified by either one of the following:
  + when the structure is carried in a region track, the entity containing the mask is a track identified by a track reference of type 'mask' from the region track to the track containing the mask. When the 'unif' brand is used, the entity can also be an image item or a mask item identified by a track reference of type 'mask' from the region track to the image item or mask item.

NOTE 3 When the 'unif' brand is used, track\_ID and item\_ID use the same value space and the track reference of type 'mask' may include either item\_ID or track\_ID values.

* + when the structure is carried in a region item, the entity containing the mask is an image item identified by an item reference of type 'mask' from the region item to the image item or mask item.

The track or image item containing the mask shall be one of the following:

* + A mask item as defined in 11.2.3 or a derived image item of a mask item.
  + An image item or a track that is encoded in monochrome format (i.e. 4:0:0 chroma format).
  + An image item or a track that is encoded in colour. In such a case, it shall be encoded in a colour format with a luma plane and chroma planes (e.g. as 4:2:0 YCbCr). Since only the luma plane is relevant, the chroma planes should be ignored.

A region may be empty if it falls entirely outside the image. An empty region should be ignored.

The mask contained in a sample of another track or in an image item (i.e., when geometry\_type equals 4) applies to the region defined by the coordinates x, y, width and height in the reference space after performing the implicit resampling of the mask caused by the difference between the size of the region documented by width and height and the size of the sample or image item containing the mask. The implicit resampling of the mask shall be performed after applying all the transformation associated with the sample or image item containing the mask, if any.

When the mask is stored as an image item or as a sample of a track, transformations associated with the image item or track shall be applied to the mask before it is used to define the geometry of a region inside the reference space.

The following semantics apply to a mask stored as a mask item, image item or sample of another track:

* the maximum sample value (e.g. 255 for 8-bit sample values) in the mask image means that the corresponding pixel in the sample of the associated track is part of the region;
* the minimum sample value (i.e. 0) in the mask image means that the corresponding pixel in the sample of the associated track is not part of the region;
* other sample values (e.g. 1 to 254 for 8-bit sample values) is representative of a probability that the corresponding pixel in the sample of the associated track is part of the region. Higher pixel values represent higher probability values.

NOTE 4 The term “sample value” used above is to be interpreted as “luma sample value” if the mask is stored as an image item or sample encoded with separate luma and chroma planes.

NOTE 5 Non-zero mask pixel values can have application-dependent semantics. For example, they can represent confidence scores for each pixel of an AI-generated mask, or saliency score indication per pixel in the defined region.

An alpha plane auxiliary track or image item may be referenced as a mask. In such case, the reference space size should be equal to the alpha plane size and the coordinates (x, y) of the region should be (0, 0).

#### Syntax

aligned (8) class RegionGeometryStruct(unsigned int version, unsigned int flags, unsigned int is\_region\_track) {  
 unsigned int field\_size = ((flags & 1) == 1) ? 32 : 16;  
 unsigned int(8) geometry\_type;  
 if (version >= 0) {  
 if (geometry\_type == 0) {  
 // point  
 signed int(field\_size) x;  
 signed int(field\_size) y;  
 }  
 else if (geometry\_type == 1) {  
 // rectangle  
 signed int(field\_size) x;  
 signed int(field\_size) y;  
 unsigned int(field\_size) width;  
 unsigned int(field\_size) height;  
 }  
 else if (geometry\_type == 2) {  
 // ellipse  
 signed int(field\_size) x;  
 signed int(field\_size) y;  
 unsigned int(field\_size) radius\_x;  
 unsigned int(field\_size) radius\_y;  
 }  
 else if (geometry\_type == 3 || geometry\_type == 6) {  
 // polygon or polyline  
 unsigned int(field size) point\_count;  
 for (i=0; i < point\_count; i++) {  
 signed int(field\_size) px;  
 signed int(field\_size) py;  
 }  
 }  
 else if (geometry\_type == 4) {   
 // referenced mask  
 signed int(field\_size) x;  
 signed int(field\_size) y;  
 unsigned int(field\_size) width;  
 unsigned int(field\_size) height;  
 if (is\_region\_track) unsigned int(field\_size)mask\_ref\_idx;  
 }  
 else if (geometry\_type == 5) {   
 // inline mask   
 signed int(field\_size) x;  
 signed int(field\_size) y;  
 unsigned int(field\_size) width;  
 unsigned int(field\_size) height;  
 unsigned int(8) mask\_coding\_method;   
 if (mask\_coding\_method != 0)   
 unsigned int(32) mask\_coding\_parameters;  
 bit(8) data[];  
 }  
 }  
}

[Editor’s note: the parameter track\_mask\_idx is currently defined only for region track. In region item, this information is implicitly determined as equal to the number of occurrences of geometries with (geometry\_type == 4) in the regions loop. Should we consider unifying the definition by defining explicitly this parameter for both region track and region item?]

#### Semantics

version is an integer that specifies the version of the format of this structure.

(flags & 1) equal to 0 specifies that the length of the fields x, y, width, height, radius\_x, radius\_y, point\_count, px, and py is 16 bits. (flags & 1) equal to 1 specifies that the length of the fields x, y, width, height, radius\_x, radius\_y, point\_count, px, and py is 32 bits. The values of flags greater than 1 are reserved.

is\_region\_track indicates whether this structure is defined in a sample of a region track or in a region item. It shall be equal to 1 when the structure is defined in a sample of a region track and it shall be equal to 0 when the structure is defined in a region item.

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 rectangle.

2: the region is described as an ellipse.

3: the region is described as a polygon.

4: the region is described as a mask defined in:

* + - a referenced image item or mask item when this structure is defined in a region item; or,
    - a referenced image item, mask item or in a sample of a referenced track when this structure is defined in a region track.

5: the region is described as a mask defined inside the data of this structure.

6: the region is described as a polyline.

Other values are reserved.

x, y specify the coordinates of the point composing the region relatively to the reference space when its geometry is a point. x, y specify the top, left corner of the region relatively to the reference space when its geometry is a rectangle or a mask. x, y specify the centre of the region relatively to the reference space when its geometry is an ellipse. The value (x = 0, y = 0) represents the position of the top-left pixel in the reference space.

NOTE 1 Negative values for the x or y fields enable to specify points, top-left corners, and/or centres that are outside the image. This can be useful for updating region annotations during the edition of an HEIF file.

width, height specify, relatively to the reference space, the width and the height of the region when its geometry is a rectangle or a mask. When geometry\_type equals 4, the value 0 indicates that the corresponding width or height value is provided by the ImageSpatialExtentsProperty associated with the item containing the mask or the width and height in the TrackHeaderBox of the track containing the mask. When geometry\_type does not equal 4, the value 0 is reserved.

radius\_x specifies, relatively to the reference space, the radius on x-axis of the region when its geometry is an ellipse.

radius\_y specifies, relatively to the reference space, the radius on y-axis of the region when its geometry is an ellipse.

point\_count is the number of points contained in a polygon or a polyline.

NOTE 2 A polygon specifying the geometry of a region is always closed and therefore there is no need to repeat the first point of the polygon as the ending point of the polygon.

px, py specify the coordinates of the points composing the polygon or the polyline relatively to the reference space. The value (px = 0, py = 0) represents the position of the top-left pixel in the reference space.

mask\_ref\_idx specifies the index of the track reference of type 'mask' referring to the track, or possibly the image item when the 'unif' brand is used, from which to retrieve the mask to apply. When a track is referenced, the sample in that track from which mask data is retrieved is the one that is temporally aligned with the current sample in the source track or the nearest preceding one in the media presentation timeline. The first track reference has the index value 1; the value 0 is reserved.

mask\_coding\_method indicates the coding method applied on the mask contained in data. The following values are defined:

0: No mask encoding scheme is applied.

1: Mask is compressed with deflate() as defined in IETF RFC 1951.

Other values are reserved.

mask\_coding\_parameters indicate additional encoding parameters needed for successfully processing the coded mask data. When mask\_coding\_method is equal to 1, mask\_coding\_parameters indicate the number of bytes in the coded mask array data. The value of mask\_coding\_parameters is reserved when the value of mask\_coding\_method is greater than 1.

data contains the coded or uncompressed representation of a mask that contains the pixels for an inline mask in raster-scan order. Each pixel is represented using a single bit and 8 pixels are packed in one byte. Byte packing shall be in big-endian order. No padding shall be put at the end of each line if the width of the mask is not a multiple of 8 pixels. Only the last data byte shall be padded with bits set to 0.

### Mask item

#### Definition

An image item with an item\_type value of 'mski' is a mask item that defines a mask. A mask item can be associated with a region item as defined in 11.2.1 to define a mask for a particular region that is defined by the region item.

A mask item has a defined width and height in pixels that shall be signalled using an ImageSpatialExtentsProperty associated with the mask item. Its data represents a lossless-compressed or uncompressed series of bits that corresponds to pixels for a mask in raster-scan order.

If a mask item’s data is lossless-compressed, the compression method shall be signalled using the content\_encoding parameter in the item’s ItemInfoEntry.

Mask specific configuration information shall be stored in MaskConfigurationProperty associated with the mask item.

bits\_per\_pixel and byte packing are defined as follows:

* when bits\_per\_pixel of MaskConfigurationProperty equals to 1, 2 or 4; pixels packed per byte are 8, 4 or 2, respectively. Byte packing shall be in big-endian order. No padding shall be put at the end of each line if the mask width is not a multiple of 8 pixels. Only the last data byte shall be padded.
* when bits\_per\_pixel of MaskConfigurationProperty equals to 8, 16 or 24; the mask value of a pixel is represented with 1, 2 or 3 bytes, respectively. Bytes of a pixel shall be serialized starting from the most significant byte.

Pixel values shall be interpreted as defined in 11.2.1.

#### Mask configuration item property

##### Definition

|  |  |
| --- | --- |
| Box type: | 'mskC' |
| Property type: | Descriptive item property |
| Container: | ItemPropertyContainerBox |
| Mandatory (per item): | Yes, for an image item of type 'mski' |
| Quantity (per item): | One, for an image item of type 'mski' |
|  |  |

Each image item of type 'mski' shall have an associated MaskConfigurationProperty.

The MaskConfigurationProperty provides information required to generate the mask of the associated mask item.

essential shall be equal to 1 for a MaskConfigurationProperty.

##### Syntax

aligned(8) class MaskConfigurationProperty  
extends ItemFullProperty('mskC', version = 0, flags = 0){  
 unsigned int(8) bits\_per\_pixel;  
}

##### Semantics

bits\_per\_pixel provides the number of bits per pixel. It shall be 1, 2, 4, 8, 16 or 24. Other values are reserved.

## Regions and region annotations for an image item

### General

A region annotation consists in metadata or image items associated with one or more regions of an image item.

A region annotation may be associated with one or more regions of an image item by:

* describing in a region item the geometry of one or more regions, as defined in 11.3.2;
* associating the region item (or a derived region item using the region item as input as defined in 11.3.3) with the image item it describes using the 'cdsc' (content describes) item reference from the region item (or derived region item) to the image item; and,
* associating any or all of the following with the region item (or the derived region item):
  + descriptive image properties, using the ItemPropertyAssociationBox;

NOTE For instance, a region annotation can use a UserDescriptionProperty to associate a description/tags with a region of an image item.

* + metadata items, using an item reference of type 'cdsc' from the metadata item to the region item.
  + image items or an entity group, using an item reference of type 'eroi' from the region item to the image item or entity group.

The region annotation applies to each region described in the region item individually.

The same region annotation may be associated with several image items by associating the same region item with multiple image items.

### Region item

#### Definition

An item with an item\_type value of 'rgan' is a region item that defines one or more regions of an image.

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

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

The geometries of the regions described by the region item are specified in the data of the region item.

The reference space is defined as a 2D coordinate system with the origin (0,0) located at the top-left corner and a maximum size defined by reference\_width and reference\_height; the x-axis is oriented from left to right and the y-axis from top to bottom.

When a region item describes several regions represented by a mask stored in an image item (i.e. when geometry\_type equals 4), there shall be one reference for each such region inside the item reference of type 'mask'. The Nth reference in the item reference of type 'mask' identifies the mask for the Nth region with geometry\_type equals to 4, in declaration order, in the region item.

NOTE 1 If the same mask stored as an image item is used for multiple regions declared in a region item, this image item will be referenced several times in the item reference of type 'mask'.

#### Syntax

aligned (8) class RegionItem {  
 unsigned int(8) version = 0;  
 unsigned int(8) flags;  
 unsigned int field\_size = ((flags & 1) + 1) \* 16; // this is a temporary, non-parsable variable  
 unsigned int(field\_size) reference\_width;  
 unsigned int(field\_size) reference\_height;  
 unsigned int(8) region\_count;  
 for (r=0; r < region\_count; r++) {  
 RegionGeometryStruct(version, flags, 0) region\_geometry;  
 }  
}

#### Semantics

version shall be equal to 0.

(flags & 1) equal to 0 specifies that the length of fields defined using field\_size is 16 bits. (flags & 1) equal to 1 specifies that the length of fields defined using field\_size is 32 bits. The values of flags greater than 1 are reserved.

reference\_width, reference\_height specify, in pixel units, the width and height, respectively, of the reference space on which the regions are placed.

region\_count specifies the number of regions defined in the region item.

region\_geometry specifies the geometry of a region as defined in 11.2.2.

### Derived region item

#### General

An item is a derived region item, when it includes a 'drgn' item reference to one or more other region items, which are inputs to the derivation. The reference space and regions defined by a derived region item are obtained by applying the operation of the derived region item to the reference space and regions of the input region items of the derived region item. The exact operation performed to obtain the reference space and the regions is identified by the item\_type of the item.

Transformative item properties associated with a derived region item shall be applied to the reference space and regions defined by the derived region item before they are applied to the image item referenced by the derived region item with an item reference of type 'cdsc'.

The number of SingleItemTypeReferenceBoxes with the box type 'drgn' and with the same value of from\_item\_ID shall not be greater than 1.

The following clauses specify the item\_type and the syntax of the item data for some derived region items.

#### Derived region item types

##### Identity derivation

A derived region item of the item\_type value 'iden' (identity transformation) may be used when it is desired to use transformative properties to derive a region item. The derived region item shall have no item body (i.e. no extents), and reference\_count for the 'drgn' item reference of a 'iden' derived region item shall be equal to 1.

NOTE 1 A derived region item of type 'iden' has an empty item body, and the one or more regions of an image defined by this derived region item is the result of applying the transformative item properties associated with this derived region item.

NOTE 2 A derived region item of type 'iden' can be used, for example, when it is desirable to have both the original version of a region item associated with an image item and a cropped version of the same region item (obtained through the 'clap' transformative item property) associated with a cropped version of the same image item.

## Regions and region annotations for an image sequence or a video track

### General

A region annotation for an image sequence or a video track may consist in metadata or images or audio samples carried in image items, samples of metadata or video or audio track, or properties from a sample group associated with one or more regions of an image carried in a sample of an image sequence or video track.

A region annotation may be associated with one or more regions of a sample of an image sequence or video track by:

* describing in a sample of a region track the geometry of these one or more regions;
* associating the region track with the image sequence or video track it describes using the 'cdsc' (content describes) track reference from the region track to the image sequence or video track;
* associating tracks or items carrying the region annotation with the region track using a track reference of type 'anot' from the region track to the image items or the tracks, and/or defining the SampleGroupDescriptionBoxes containing the region annotation; and
* defining a sample-to-region-id-mapping sample grouping ('regm') in the region track providing the mapping between the region identifiers and the region annotations.

The region annotation applies to each region described in time-aligned sample of the region track individually.

### Region track

#### Definition

A metadata track with a sample entry 'rgan' is a region track. Samples of a region track define one or more regions inside images carried in samples of an associated image sequence or video track (also denoted source track in the following).

The region track is associated with the source track using a track reference of type 'cdsc' from the region track to the source track.

A region track allows associating a region annotation with each region. A region annotation may be an item (e.g., an image item, a metadata item), samples from another track, or a SampleGroupDescriptionEntry.

For each sync sample of the source track, the region track should have a time-aligned sample describing all regions of this sync sample.

Each region is associated with a region identifier. The scope of region identifiers is indicated by the persistent\_region\_ids flag in the RegionTrackConfigBox of the sample entry. The region identifier can be used to associate an annotation with a region in a sample via an association map defined in a sample-to-region-id-mapping sample group. In such case, region identifier values in samples and in the sample group have the same value space.

Different regions in a same sample or in consecutive samples may have the same region identifier. When multiple regions in the same sample have the same region identifier, this indicates that each of these regions is associated with the same set of region annotations.

A SampleGroupDescriptionEntry should only be associated with a region defined by a region track when the property value for the region differs from the matching property value, explicit or implicit, for the whole sample of the associated source track.

The geometries of the regions described in the region track are specified in the data of the samples of the region track.

The media presentation timeline is used to conclude correspondence of samples of the region track in relation to samples of the source track. The presentation times of the samples of a source track may or may not be aligned with the presentation times of the samples of the associated region track. Regions that apply to a given sample of a source track are regions defined in the sample of the associated region track that is active at the beginning of the presentation time of the given sample of the source track.

The reference space is defined as a 2D coordinate system with the origin (0,0) located at the top-left corner and a maximum size defined by reference\_width and reference\_height in sample entry; the x-axis is oriented from left to right and the y-axis from top to bottom.

#### Sample entry

##### Definition

|  |  |
| --- | --- |
| Sample entry type: | 'rgan' |
| Container: | SampleDescriptionBox |
| Mandatory: | Yes |
| Quantity: | One or more sample entries may be present |
|  |  |

The sample entry documents in a RegionTrackConfigBox the configuration of the region sample and the reference space inside which regions are defined.

The version in RegionTrackConfigBox indicates both the version of the box and the version of the sample format.

The following flags are defined for the flags field of RegionTrackConfigBox:

0x000001 field\_length\_size; indicates the length of various fields. When set, it specifies that the length of a field is 32 bits. Otherwise, the length of a field is 16 bits.

0x000002 persistent\_region\_ids; indicates the scope of region identifiers. When set, it specifies that the scope of region identifiers is persistent over samples, i.e., a same region identifier in different samples mapped to this sample entry identifies the same region. Otherwise, the scope of region identifiers is the sample.

NOTE A scope of region identifiers persistent over samples is useful for tracking regions. On contrary, a scope of region identifiers limited to the sample is useful when splicing separate contents that use the same region identifiers and can reduce the description cost when multiple regions in a sample are associated with the same set of annotations.

##### Syntax

aligned(8) class RegionTrackConfigBox () extends FullBox ('rgaC', version=0, flags) {  
 unsigned int((field\_length\_size + 1) \* 16) reference\_width;  
 unsigned int((field\_length\_size + 1) \* 16) reference\_height;  
}

aligned(8) class RegionSampleEntry  
extends MetadataSampleEntry ('rgan'){  
 RegionTrackConfigBox config; // mandatory  
}

##### Semantics

reference\_width, reference\_height specify, in pixel units, the width and height, respectively, of the reference space inside which the regions are placed.

#### Sample format

##### Definition

This subclause defines the sample format for region track. A sample of a region track defines one or more regions.

##### Syntax

aligned (8) class RegionSample {  
 unsigned int field\_size = ((RegionTrackConfigBox.flags & field\_length\_size) == field\_length\_size) ? 32 : 16;  
// this is a temporary, non-parsable variable  
 unsigned int(32) region\_count;  
 for (r=0; r < region\_count; r++) {  
 unsigned int(32) region\_identifier;  
 RegionGeometryStruct(RegionTrackConfigBox.version, RegionTrackConfigBox.flags, 1) region\_geometry;  
 }  
}

##### Semantics

RegionTrackConfigBox indicates the configuration box in the matching sample entry.

region\_count specifies the number of regions defined in the sample.

region\_identifier specifies the identifier of the region.

region\_geometry specifies the geometry of a region as defined in 11.2.2.

### Sample groups for region track

#### Sample-to-region-id-mapping sample group

##### Definition

The sample-to-region-id-mapping ('regm') sample grouping provides a flexible way to associate regions inside samples of a region track, as defined in 11.4.2, with region annotations, as defined in 11.4.1. It provides the mapping between region identifiers in samples of the region track and region annotations carried in various types of containers: tracks, items or sample group descriptions.

The sample-to-region-id-mapping sample grouping should only be present in a region track, and may be ignored otherwise.

The number of entries (entry\_count) declared in a SampleToRegionIdMappingEntry does not necessarily match 1:1 with the number of regions defined in associated samples of the region track. When there is no regionID value for an identifier of a region declared in a sample, this means that no annotation is associated with this region for the duration of this sample. Conversely, when there is no region declared in a sample with same value of identifier as a regionID value, this means that the corresponding annotations are not associated with any regions for the duration of this sample.

##### Syntax

class SampleToRegionIdMappingEntry()  
extends SampleGroupDescriptionEntry('regm') {  
 unsigned int(32) entry\_count;  
 for (i=1; i<= entry\_count; i++) {  
 unsigned int(32) regionID;  
 unsigned int(16) association\_count;  
 for (i=0; i<association\_count; i++) {  
 unsigned int(5) reserved;  
 unsigned int(3) annotation\_container\_type;  
 unsigned int(32) annotation\_reference\_type;  
 unsigned int(32) annotation\_idx;  
 }  
 }  
}

##### Semantics

entry\_count is an integer that gives the number of entries in the mapping table, i.e., the number of regions in the mapping table.

regionID is an integer that specifies the identifier of the region.

annotation\_count is an integer that specifies the number of annotations associated with the region.

annotation\_container\_type specifies the type of container carrying the annotation associated with the region. Following values for annotation\_container\_type are defined:

0: the annotation is carried in samples of the track or in the item identified by annotation\_reference\_type and annotation\_idx in the track reference box. When the annotation is carried in samples of the track identified by annotation\_reference\_type and annotation\_idx, the sample in that track temporally aligned, or nearest preceding in the media presentation timeline, comprises the annotation applying to the corresponding sample of the track referenced by the region track with a track reference 'cdsc'.

1: the annotation is the SampleGroupDescriptionEntry corresponding to the index annotation\_idx in the SampleGroupDescriptionBox with grouping\_type equal to annotation\_reference\_type.

2-7: reserved.

annotation\_reference\_type specifies the four-character code type of the container carrying the annotation depending on the annotation\_container\_type value. If the value of annotation\_container\_type equals 0, it specifies the reference\_type of the TrackReferenceTypeBox in the TrackReferenceBox containing the identifier of the entity (i.e., track or item) containing the annotation associated with the region. If the value of annotation\_container\_type equals 1, it specifies the grouping\_type of the SampleGroupDescriptionBox containing the annotation associated with the region. The meaning of annotation\_reference\_type is undefined for other values of annotation\_container\_type.

annotation\_idx specifies the index to retrieve the container carrying the annotation depending on the annotation\_container\_type value. If the value of annotation\_container\_type equals 0, it specifies the index of the track reference of type annotation\_reference\_type identifying the track or item from which to retrieve the annotation associated with the region. If the value of annotation\_container\_type equals 1, it specifies the index of the SampleGroupDescriptionEntry in the SampleGroupDescriptionBox with grouping\_type equal to annotation\_reference\_type from which to retrieve the annotation associated with the region. The meaning of annotation\_idx is undefined for other values of annotation\_container\_type.



*Annex H.2*

Replace the following in subclause H.2.1 (“*Definition*”)

The concatenation of the contents of the optional JPEG configuration box (the JPEGprefix bytes) with the extents of the JPEG image item shall conform to the specification for a JPEG compressed image as defined in ISO/IEC 10918-1, starting with the SOI (start of image) marker and ending with the EOI (end of image) marker.

With

The concatenation of the contents of the optional JPEGConfigurationProperty (the JPEGprefix bytes) with the extents of the JPEG image item shall conform to the specification for a JPEG compressed image as defined in ISO/IEC 10918-1, starting with the SOI (start of image) marker and ending with the EOI (end of image) marker.

Move the content of subclause H.2.2 (“*JPEG configuration item property*”) in a new subclause H.2.2.1 (“*Definition*”) under H.2.2 (“*JPEG configuration item property*”)

Add the following in new subclause H.2.2.1 (“Definition”)

JPEGConfigurationProperty contains an initial part of a JPEG compressed image as defined in ISO/IEC 10918-1, starting with the SOI (start of image) marker.

NOTE JPEGConfigurationProperty can be used to share quantization and other tables across several images.

Renumber the subclause H.2.3 (“*Syntax*”) to H.2.2.2 (“Syntax”) and in this subclause, rename the class JPEGConfigurationBox into JPEGConfigurationProperty.

Add a new subclause H.2.2.3 (“*Semantics*”) containing the following semantics:

JPEGprefix is the content to concatenate before the data of the associated image item.

Renumber subclause H.3.2 (“*Derivation from ISO/IEC 14496-12*”) into H.3.3 and add a new subclause H.3.2 (“*JPEG configuration box*”) as follows:

H.3.2 JPEG configuration box

H.3.2.1 Definition

|  |  |
| --- | --- |
| Box type: | 'jpgC' |
| Container: | Sample entry for a JPEG image sequence |
| Mandatory: | No |
| Quantity: | Zero or one |
|  |  |

JPEGConfigurationBox contains an initial part of a JPEG compressed image as defined in ISO/IEC 10918-1, starting with the SOI (start of image) marker.

NOTE JPEGConfigurationBox can be used to share quantization and other tables across several images.

H.3.2.2 Syntax

class JPEGConfigurationBox extends Box('jpgC') {  
 unsigned int(8) JPEGprefix[];  
}

H.3.2.3 Semantics

JPEGprefix is the content to concatenate before the data of each sample associated with the sample entry.

*Annex L.2.2.1*

Replace entire NOTE 3 in subclause L.2.2.1.2 (“*Image item of type 'vvc1'*”) with the following:

NOTE 3 ISO/IEC 23090-3 requires that a VVC decoder conforming to a profile at a specific level and specific tier is able to decode the first picture of a VVC bitstream when all of the following applies:

* at least one of the following:
  + the VVC decoder conforms to the Main 10 Still Picture profile and the bitstream conforms to the Main 10 profile;
  + the VVC decoder conforms to the Main 10 4:4:4 Still picture profile and the bitstream conforms to the Main 10 profile or the Main 10 4:4:4 profile;
  + the VVC decoder conforms to the Main 12 Still Picture profile and the bitstream conforms to one of the Main 10, Main 10 4:4:4, Main 12, or Main 12 Intra profiles;
  + the VVC decoder conforms to the Main 12 4:4:4 Still Picture profile and the bitstream conforms to one of the Main 10, Main 10 4:4:4, Main 12, Main 12 Intra, Main 12 4:4:4, or Main 12 4:4:4 Intra profiles;
  + the VVC decoder conforms to the Main 16 4:4:4 Still Picture profile and the bitstream conforms to one of the Main 10, Main 10 4:4:4, Main 12, Main 12 Intra, Main 12 4:4:4, Main 12 4:4:4 Intra, Main 16 4:4:4, or Main 16 4:4:4 Intra profiles;
* the VVC bitstream conforms to a tier that is lower than or equal to the specified tier;
* the VVC bitstream conforms to a level that is not level 15.5 and is lower than or equal to the specified level;
* the first picture of the VVC bitstream is an IRAP picture or a GDR picture with ph\_recovery\_poc\_cnt equal to 0, is in an output layer, and has ph\_pic\_output\_flag equal to 1.

*Annex K*

*Replace entire annex K with the following:*

1. (informative)  
     
   Examples of progressive decoding, rendering and refinement
   1. Overview

Progressive rendering refers to displaying image content in successive steps where each step improves the perceived image quality over that of the previous step and is superimposed over the image content of the previous step in the same displaying window. A progressive rendering step can improve the perceived image quality over the complete image as a whole, or region by region resulting in a region-wise progressive rendering. In the latter case, the first rendering step results into a base quality image and successive rendering steps improve the perceived image quality region by region. In a variation of region-wise progressive rendering, several initial rendering steps provide complete images, each improving the picture quality over the previous one, followed by successive rendering steps improving the perceived image quality region by region. Furthermore, several passes of region-wise progressive rendering scans of the image can be carried out, each pass with an improved picture quality over the previous pass.

Progressive decoding refers to decoding a bitstream with a single decoder instance in successive steps where each step improves the perceived image quality over that of the previous step. Progressive decoding, when enabled in a file, can be used with progressive rendering.

Progressive refinement refers to progressive rendering of the image content in a file while downloading the file.

This annex presents examples of content encoding, file structures and player operations for progressive rendering, progressive decoding and progressive refinement. This annex is organized as follows:

* Subclause K.2 contains suggestions and remarks for creating files suitable for progressive refinement. These suggestions and remarks apply to any subsequent example for progressive refinement.
* Subclause K.3 includes a simple example to use the progressive rendering entity group specified in 6.8.10 for a coded image and its thumbnail.
* Subclause K.4 describes a region-wise progressive rendering with independently coded image items. It demonstrates the use of the ProgressiveDerivedImageItemInformationProperty item property specified in 6.5.37.
* Subclause K.5 describes how progressive decoding operation can be achieved with an overlay derived image item represented by a single bitstream with temporal inter prediction. It demonstrates the use of the SingleStreamProperty item property specified in 6.5.38.
* Subclause K.6 describes the use of multi-layer images with the progressive rendering entity group.
* Subclause K.7 describes a potential player operation for progressive rendering or refinement for HEIF files.
  1. Remarks on creating a file suitable for progressive refinement

When creating a file suitable for progressive refinement, the following ordering of boxes is suggested: The file-level MetaBox precedes the MediaDataBox(es), if any. When image item(s) are contained in the ItemDataBox, the ItemDataBox is arranged to be the last box within the containing MetaBox. When an image sequence or video track is intended to serve as rendering step following image item(s), the MovieBox precedes the MediaDataBox(es) containing the samples of the track.

The coded data for entities used for progressive refinement is suggested to have an order within the file that results into progressive rendering when the player decodes and displays the coded data in its appearance order within the file.

The progressive application brand ('MiPr') of ISO/IEC 23000‑22 specifies a set of constraints for HEIF files suitable for progressive refinement. The HEIF files described in the examples of this annex for progressive refinement can additionally follow the constraints of the 'MiPr' brand, in which case the 'MiPr' can be included in the FileTypeBox.

* 1. Progressive rendering and refinement with independently coded image items

The progressive rendering entity group signals a set of image items that can be used for progressive rendering and are ordered in the file in a manner suitable for progressive refinement.

In a simple example, a file contains a primary item that is a coded image item and its thumbnail image item. The coded image data for the thumbnail image item precedes the coded image data for the primary item. The file contains a progressive rendering entity group that lists the thumbnail image item and the primary item.

* 1. Region-wise progressive rendering and refinement with independently coded image items

The ProgressiveDerivedImageItemInformationProperty describes progressive rendering steps associated with a derived image item. Each progressive rendering step specifies a count of input image items to be used as a rendering step of the derived image item and is described as a difference from the previous step.

A single source image is encoded into multiple bitstreams as follows:

* One or more base quality image bitstreams, each improving the picture quality over the previous one, are encoded using any of the following ways:
  + The source image is encoded with a base image quality. In many codecs, the image quality can be adjusted with a configurable quantization parameter.
  + The source image is low-pass filtered prior to encoding.
  + The source image is downsampled prior to encoding to obtain a smaller resolution version of the source image.
* The source image is split into regions, for example along a regular grid or based on detecting regions of interest in the source image. Each region is encoded as an independent bitstream.

Given all the bitstreams as input, a HEIF file is created as follows:

* An image item is formed from each bitstream.
* Either of the following approaches can be used to indicate region-wise progressive rendering:
  + A grid derived image item can be used with the ProgressiveDerivedImageItemInformationProperty item property to indicate the number of input images to be used in each progressive rendering steps of the image grid. A progressive rendering entity group is created with inputs being the base quality image items in ascending picture quality order, followed by the grid derived image item. The processing of the progressive rendering entity group in a player causes successive displaying of the base quality images at ascending picture quality on the displaying window, followed by the region-wise progressive rendering of the grid derived image item.
  + An overlay derived image item can be used with the ProgressiveDerivedImageItemInformationProperty item property. One or more first input images of the overlay derived image provide the base quality images in ascending picture quality order, while the subsequent input images provide region-wise progressive rendering steps. In a variation, the overlay derived image item can be also included in a progressive rendering entity group.

An example of region-wise progressive rendering is illustrated in Figure K.1. In this example, a base quality image is progressively refined starting from the top-left and ending at the bottom-right. This HEIF file contains the following elements:

* A base quality image item, indicated with an entire picture area covered by white rectangles in Figure K.1.
* 12 coded image items at improved picture quality, each indicated with a green rectangle in Figure K.1. Each of these coded image items represents a different region in a 4×3 grid.
* An overlay derived image item that uses the base quality image item and the 12 image items at improved picture quality in a diagonal scan order from the top-left corner of the grid towards the bottom-right corner.
* The overlay image item is associated with a ProgressiveDerivedImageItemInformationProperty item property.

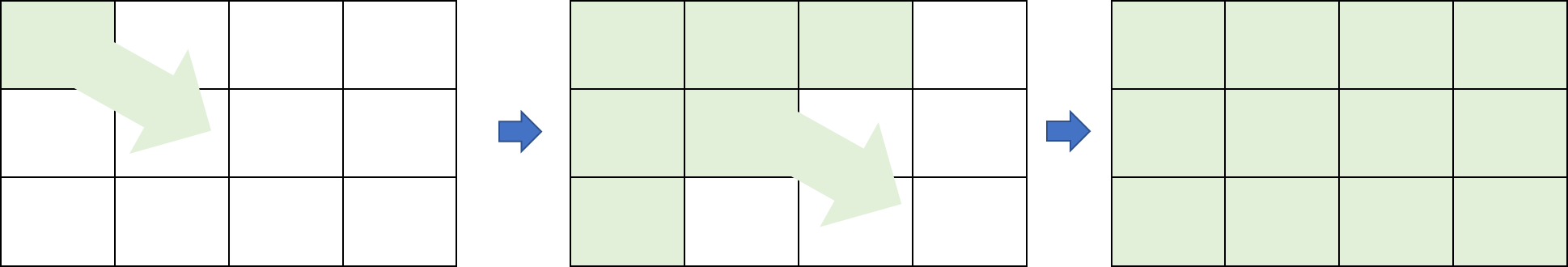


Figure K.1 — Three selected rendering steps of an exemplary region-wise progressive rendering arrangement

* 1. Overlay derived image item represented by a single bitstream with temporal inter prediction

This subclause describes how progressive decoding and rendering are achieved by encoding a single source image as multiple pictures in a single video bitstream.

A single source image can be encoded as a video bitstream that provides progressive decoding capability in any of the following ways:

* Quality-wise progressive rendering can be achieved in either of the following ways:
  + The source image is encoded with a base image quality as the first picture in the bitstream, and the same source image is encoded in one or more enhanced image qualities in respective inter-coded pictures into the same bitstream. In many codecs, the image quality can be adjusted with a configurable quantization parameter.
  + The source image is low-pass filtered prior to encoding to obtain one or more smoother versions of the source image. The smoothest low-pass filtered image is encoded as the first picture in the bitstream. Moreover, the other low-pass filtered versions of the source image, if any, are encoded as inter-coded pictures in ascending order of fidelity into the same bitstream. Finally, the source image itself is encoded as an inter-coded picture into the same bitstream.
* Resolution-wise progressive rendering:

Some video codecs, such as VVC, provide the reference picture resampling (RPR) capability where a reference picture for inter prediction can have a different spatial resolution than a picture being predicted and consequently the reference picture is resampled for the inter prediction process. The RPR feature can be used for progressive decoding as described in the next paragraph.

The source image is downsampled prior to encoding to obtain one or more smaller resolution versions of the source image. The lowest resolution image is encoded as the first picture in the bitstream, and other versions of the source image, each providing a higher resolution, are encoded as inter-coded pictures into the same bitstream.

* Region-wise progressive rendering:

Some video codecs, such as VVC, provide the capability to indicate the spatial alignment and correspondence of pictures in relation to each other. In VVC, this is achieved through indicating or inferring a scaling window for each picture, where the scaling windows specify a respective spatial area in the pictures.

A version of the source image is encoded with a base image quality or the lowest spatial resolution as the first picture in the bitstream, as described above. The source image is split into regions, for example along a regular grid or based on detecting regions of interest in the source image. Each region is encoded as an inter-coded picture into the bitstream, with inter prediction from the first picture in the bitstream.

Given a bitstream encoded by any means described above as input, a HEIF file is created as follows:

* An image item is formed from each picture of the bitstream. Consequently, the first picture in the bitstream becomes a conventional coded image item and each inter-coded picture becomes a predictively coded image item.
* An overlay derived image item is formed from the coded image items, where the order of pictures in the bitstream determines the layering order of the respective input image items.
* A ProgressiveDerivedImageItemInformationProperty item property is created and associated with the overlay derived image item. Each input image item of the overlay derived image item is assigned as its own rendering step in the ProgressiveDerivedImageItemInformationProperty item property.
* A SingleStreamProperty item property is created and associated with the overlay derived image item. The SingleStreamProperty item property indicates that the input image items to a derived image item, when concatenated, form a single bitstream that is conformant to the coding format of the input image items and is decodable with a single decoder.

An example of region-wise progressive rendering is provided in Figure K.2 and described in this paragraph. An overlay derived image item is constructed from image items A to F. Image item A is a lower resolution version of a source image and image items B to F are predictively coded from image item A. The predictively coded image items B to F form the higher resolution version of the same source image divided into different regions. Spatial correspondence is indicated with dashed rectangles, which can be assigned as the scaling windows in VVC encoding. Image items A to F are input to an overlay derived image item.

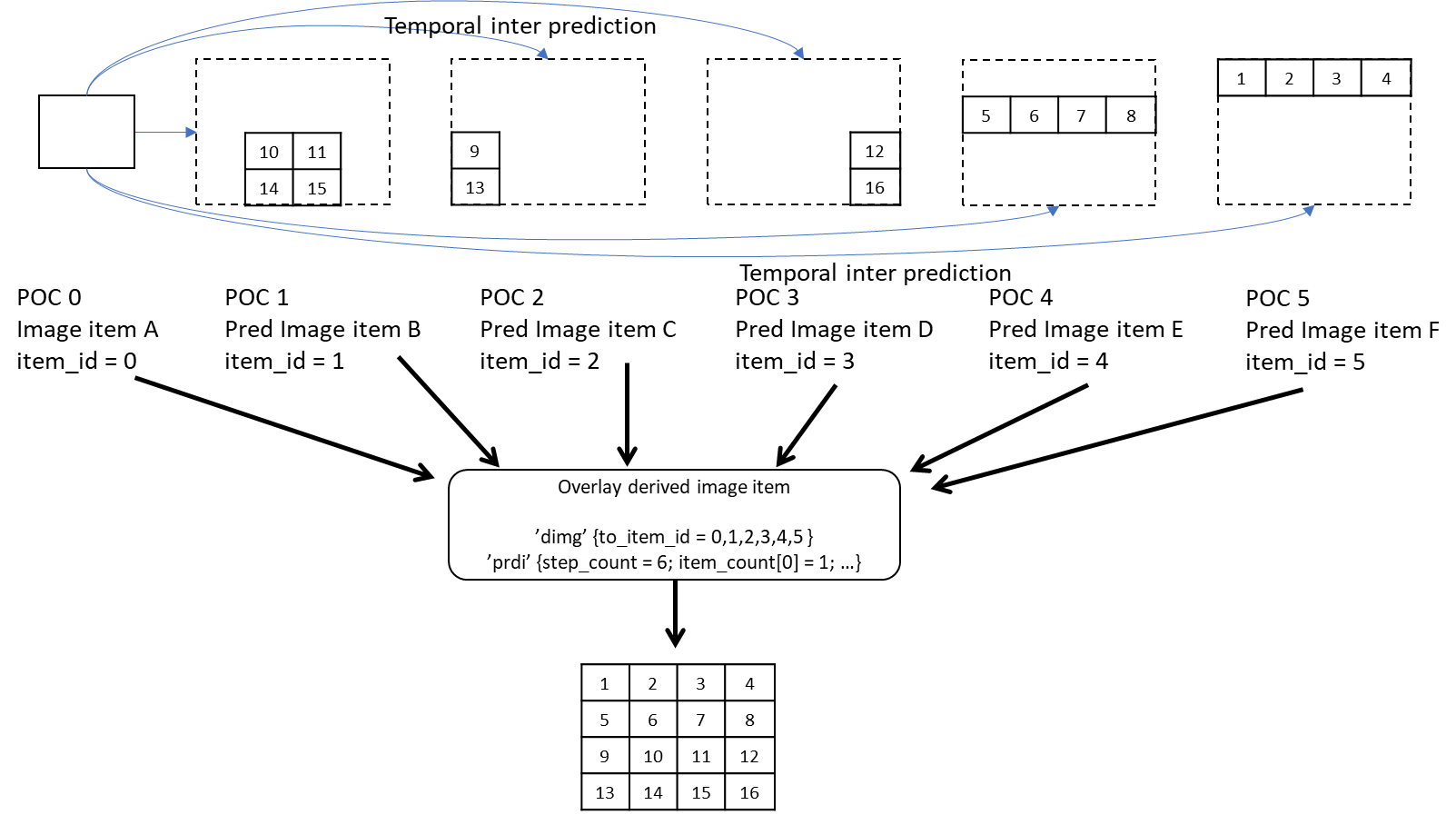


Figure K.2 — Example of progressive decoding and region-wise progressive rendering

* 1. Multi-layer images with the progressive rendering entity group

This subclause describes how progressive decoding, rendering and refinement are achieved by encoding a single source image as a multi-layer bitstream.

A single source image is encoded as a bitstream that contains a single access unit with a base layer and *N* quality or spatial enhancement layers, where *N* is greater than or equal to 1. Each enhancement layer is inter-layer-predicted from the base layer or the lower enhancement layer(s), if any.

A HEIF file is created from the bitstream as follows:

* An image item is created to contain the base layer picture. Furthermore, for each enhancement layer *M*, an image item is created containing the enhancement layer *M*, the base layer and the lower enhancement layers that are needed for decoding enhancement layer *M*, if any.

Coded pictures are stored in the HEIF file only once, since image items can be composed of extents and hence different image items can refer to the same byte range containing a coded picture of a particular layer. Coded pictures are stored in their decoding order to make the HEIF file suitable for progressive refinement.

* A TargetOlsProperty item property is created and associated with each image item, which enables to invoke the decoder appropriately. The TargetOlsProperty item property provides the output layer set index to be used as input for the decoding process of the associated coded image item.
* In a typical setting for encoding, each target output layer set is associated with a single output layer, which of the highest layer of the target output layer set. Hence, decoding an image item would implicitly produce a single reconstructed image and creation of LayerSelectorProperty item properties into the file would not be needed.
* All the image items containing different output layer sets are included in the same progressive rendering entity group ('prgr') to indicate that they can be progressively rendered. An image item that contains only the base layer is the first item in the progressive rendering entity group, followed by image items in ascending order of enhancement until the image item contain all the layers.
* All the image items are also included in the same 'altr' entity group. An image item containing all the layers is the first item in the 'altr' entity group to indicate that it is the most preferred to be displayed, followed by image items in descending order of enhancement until the image item with containing only the base layer.

A player can operate as follows:

* The player concludes from the progressive rendering entity group ('prgr') that the file, and specifically the image items in the progressive rendering entity group, are suitable for progressive refinement.
* If a decoder does not provide access to decoded pictures in non-output layers, the player decodes each image item in the progressive rendering entity group with a separate decoder instance. Each image item, represented by its output image, is used as a rendering step in progressive rendering.

Otherwise, the player concludes from the extents that all image items of the progressive rendering entity group are parts of the same scalable bitstream, decodes the entire scalable bitstream (which is included in the last image item of the progressive rendering entity group) with a single decoder instance, and uses each decoded picture as a rendering step in progressive rendering.

* 1. Player operation for progressive rendering or refinement for HEIF files

This subclause describes potential player operation for players that use progressive rendering or refinement for displaying media. Support for progressive rendering or refinement is explicitly signalled in a file by the presence of a progressive rendering entity group ('prgr'). In the absence of such a progressive rendering entity group, a player may use an 'altr' entity group as the basis for progressive rendering or refinement.

A player can get input, for example through user interaction, indicating which image item or track is to be displayed. If no such input is available, the player selects the primary item to be displayed.

K.7.1 Explicit signalling for progressive rendering or refinement

If the file contains an entity group of type 'prgr' that includes the image item or track to be displayed, the images items or tracks in that entity group are potential entities for progressive rendering or refinement.

[Ed. Note (FM): The definition of 'prgr' entity group currently forbids tracks, while 'altr' entity group authorizes both items and tracks. Is there any reason not to allow tracks in 'prgr' entity group?]

If the file contains a ProgressiveDerivedImageItemInformationProperty item property associated with a derived image item to be displayed or with a potential entity for progressive rendering or refinement, this item property indicates progressive rendering steps for the derived image item that are particularly suitable for progressive rendering or refinement. Each progressive rendering step indicated for the derived image item represents a potential entity suitable for progressive rendering or refinement. The list of input image items corresponding to a progressive rendering step can be retrieved from the number of input image items for this progressive rendering step as indicated in the ProgressiveDerivedImageItemInformationProperty item property.

K.7.2 Implicit signalling for progressive rendering or refinement

If the file does not contain an entity group of type 'prgr', but contains an entity group of type 'altr' that includes the image item or track to be displayed, the image items or tracks in that entity group are regarded as potential entities for progressive rendering or refinement. Otherwise, if the image item or track to be displayed has an associated thumbnail image item or track, both are regarded as potential entities for progressive rendering or refinement. Otherwise, only the image item or track to be displayed is regarded as a potential entity for progressive rendering or refinement.

NOTE The semantics of the 'altr' entity group recommend a player to select the first entity from the list of entity\_id values that it can process (e.g. decode and play for mapped items and tracks that are part of the presentation) and that suits the application needs.

K.7.3 Progressive rendering or refinement steps

The player determines the dependencies, if any, of the potential entities for progressive rendering or refinement. For example, for a derived image item that is a potential entity for progressive decoding or refinement, the player identifies all the directly and indirectly required input image items.

The player determines the next potential entity for progressive rendering or refinement that can be displayed, taking into account the entities on which the potential entity depends. A potential entity can be displayed if the potential entity and all the entities it depends on have been fully received.

When the potential entity has an associated ProgressiveDerivedImageItemInformationProperty item property it can be displayed if:

* The potential entity itself (i.e., the data associated to the potential entity) has been fully received;
* A subset of the entities it depends on corresponding to a progressive rendering step described by the ProgressiveDerivedImageItemInformationProperty item property has been fully received.

If a potential entity for progressive rendering or refinement can be displayed, the player determines if this potential entity enhances the currently displayed entity, if any, as follows:

* If no entity has been displayed so far, the player displays the potential entity for progressive rendering or refinement.
* Otherwise, if a progressive rendering entity group ('prgr') was identified above and the potential entity for progressive rendering or refinement appears later in the list of the progressive rendering entity group than the currently displayed entity and does not appear later in the list than the entity to display, the potential entity for progressive rendering or refinement is displayed.
* Otherwise, if an 'altr' entity group was identified above and the potential entity for progressive rendering or refinement appears earlier in the list of entity\_id values of the 'altr' entity group than the currently displayed entity, the potential entity for progressive rendering or refinement is displayed.
* Otherwise, if the potential entity for progressive rendering or refinement is the master image of a thumbnail image being displayed currently or the master image sequence of a thumbnail image sequence being displayed currently, the potential entity for progressive rendering or refinement is displayed.
* Otherwise, if a ProgressiveDerivedImageItemInformationProperty item property is associated with the potential entity for progressive rendering or refinement and if a progressive rendering step described by the ProgressiveDerivedImageItemInformationProperty item property can be displayed and if that progressive rendering step is listed after the currently displayed progressive rendering step inside the ProgressiveDerivedImageItemInformationProperty item property, then the potential entity for progressive rendering or refinement is displayed,
* Otherwise, the potential entity for progressive rendering or refinement is not considered as an enhancement of the currently displayed entity and is hence not displayed.

Possibly, a previous potential entity may be used to fill the blanks left by the progressive rendering step of an entity associated with a ProgressiveDerivedImageItemInformationProperty item property.

The displaying of an image item used for progressive rendering or refinement can be done by decoding and displaying the image item in spatially progressive manner. The displaying of an image sequence track used for progressive rendering or refinement can utilize progressive downloading, i.e. the playback can start before the samples of the track are entirely received.

Some decoders may be able to decode parts of coded pictures out of their normative decoding order specified in the coding format. For example, HEVC decoders may be able to decode slices of a coded pictures out of their normative decoding order. Such decoders may decode extents of a coded image item in the order they appear in the file, even if that order were not the normative decoding order.