ISO/IEC JTC 1/SC 29/WG 03 N0465

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

**Document type:** Output Document

**Title:** Technology under Consideration on ISO/IEC 23008-12

**Status:** Approved

**Date of document:** 2022-01-26

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

**No. of pages:** 4 (with cover page)

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

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

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

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

**ISO/IEC JTC 1/SC 29/WG 03 N0465**

**January 2022, Virtual**

|  |  |
| --- | --- |
| **Title** | **Technology under Consideration on ISO/IEC 23008-12** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **21181** |

# Abstract

This document collects following candidate technologies for the High Efficiency Image File Format (HEIF) (ISO/IEC 23008-12).

Table of Contents

[Abstract 1](#_Toc94105130)

[1. Item properties for camera intrinsic and extrinsic matrices 2](#_Toc94105131)

**[1](#_Toc94105132)****[Camera extrinsic matrix property box](#_Toc94105132)** [2](#_Toc94105132)

**[1.1 Definition](#_Toc94105133)** [2](#_Toc94105133)

**[1.2](#_Toc94105134)****[Syntax](#_Toc94105134)** [3](#_Toc94105134)

**[1.3](#_Toc94105135)****[Semantics](#_Toc94105135)** [3](#_Toc94105135)

**[1.4](#_Toc94105136)****[Discussion](#_Toc94105136)** [4](#_Toc94105136)

**[1.5](#_Toc94105137)****[Considerations for timed metadata](#_Toc94105137)** [4](#_Toc94105137)

**[2](#_Toc94105138)****[Camera intrinsic matrix property box](#_Toc94105138)** [6](#_Toc94105138)

**[2.1](#_Toc94105139)****[Definition](#_Toc94105139)** [6](#_Toc94105139)

**[2.2](#_Toc94105140)****[Syntax](#_Toc94105140)** [6](#_Toc94105140)

**[2.3](#_Toc94105141)****[Semantics](#_Toc94105141)** [6](#_Toc94105141)

**[2.4](#_Toc94105142)****[Discussion](#_Toc94105142)** [7](#_Toc94105142)

**[3](#_Toc94105143)****[Alternatives considered](#_Toc94105143)** [7](#_Toc94105143)

[2. Carriage of Text Items 7](#_Toc94105144)

**[1](#_Toc94105145)****[Introduction](#_Toc94105145)** [8](#_Toc94105145)

**[2](#_Toc94105146)****[Proposal](#_Toc94105146)** [8](#_Toc94105146)

[3. Region annotations for image sequence or video tracks 13](#_Toc94105147)

**[1](#_Toc94105148)****[Motivation](#_Toc94105148)** [13](#_Toc94105148)

**[2](#_Toc94105149)****[Proposal](#_Toc94105149)** [14](#_Toc94105149)

**[3](#_Toc94105150)****[Discussion](#_Toc94105150)** [15](#_Toc94105150)

[4. Matrix-based transformation for image items 15](#_Toc94105158)

[5. Signaling for pre-derived coded image items 16](#_Toc94105159)

# Item properties for camera intrinsic and extrinsic matrices

*[ Ed. (FD): From MPEG#137 minutes => consider adding to amendment at the next meeting, after discussion and revision in the meeting]*.

This topic was discussed as [HEIF Issue#62](http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/HEIF/-/issues/62) on the MPEG GitLab.

1. **Camera extrinsic matrix property box**

There are several specifications in the MPEG-I project that signal camera extrinsics (see for example 23090-12 MPEG Immersive Video (MIV)). **This proposal uses almost the same definition and semantics, with the difference that integers are used rather than floating point**. See section 1.4 for a discussion of why this is preferable for still images.

**1.1 Definition**

Box Type: 'cmex'  
Container: ItemPropertyContainerBox (‘ipco’)  
Mandatory (per item): No  
Quantity (per item): Zero or one

The CameraExtrinsicsMatrix allows writers to communicate the spatial setup for a set of cameras. It is specified in the form of x, y, and z coordinates as well as an orientation in quaternion representation.

The coordinate system of the fields refers to the scene in which all the camera systems are positioned and can be identified, when 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 systems. It is recommended that the origin is placed either at one of the cameras in a multi-camera setup, or in the geometric center.

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

* Simple stereo pair (typically only requires a baseline)
  + Use flags value 0x000001
* Multiple cameras facing same direction (typically requires position in 2 or 3 axes)
  + Use flags values from 0x000001 to 0x000007 (depending on which axes are non-zero)
* Panorama image collections (typically only requires orientation)
  + Use flags value 0x000008
* 3D mapping image collections (typically requires both 3D position and orientation)
  + Use flags value 0x00000F
  1. **Syntax**

aligned(8) class CameraExtrinsicsMatrix  
extends ItemFullProperty('cmex', version = 0, **flags**) {  
 if (flags & 0x1) {  
 signed int(32) pos\_x;  
 }  
 if (flags & 0x2) {  
 signed int(32) pos\_y;  
 }  
 if (flags & 0x4) {  
 signed int(32) pos\_z;  
 }  
 if (flags & 0x8) {  
 signed int(16) quat\_x;  
 signed int(16) quat\_y;  
 signed int(16) quat\_z;  
 }  
 if (flags & 0x10) {  
 unsigned int(32) id;  
 }  
}

* 1. **Semantics**

**flags**: is a 24-bit integer with flags; the following values are defined:

position\_x: Flag mask is 0x000001. The value 1 indicates that the position is signalled along the x axis.

position\_y: Flag mask is 0x000002. The value 2 indicates that the position is signalled along the y axis.

position\_z: Flag mask is 0x000004. The value 4 indicates that the position is signalled along the z axis.

orientation: Flag mask is 0x000008. The value 8 indicates that orientation is signalled.

id\_present: Flag mask is 0x000010. The value 16 indicates that the coordinate system id is signalled.

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 -214 to 214, inclusive. 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 -214 to 214, inclusive. 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 -214 to 214, inclusive. When not present, its value shall be inferred to be 0.

id: Specifies the coordinate system id. When not present, its value shall be inferred to be 0. CameraExtrinsicsMatrix instances with the same id indicate that the associated image items were captured in the same coordinate system.

The values of the quaternion representation are computed as follows:

qX = **quat\_x** / 214  
qY = **quat\_y** / 214  
qZ = **quat\_z** / 214

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 – ( qX2 + qY2 + qZ2 ) )

* 1. **Discussion**

For images, re-encoding, transcoding, and derivative creation are all common operations. Converting metadata to/from floating-point will be almost guaranteed to cause rounding errors, which we want to avoid when possible. For the camera extrinsics matrix, the units are in scene/real-world coordinates and it is therefore relatively easy to select a fixed denominator (µ or 1000000) that allows sufficient precision for all use-cases.

For this reason, an integer-based representation has been used in this item property. Using an integer representation of the properties also matches what is done for other HEIF item properties.

Using µm as the unit combined with a 32-bit signed integer allows specifying a maximum spatial displacement in one axis of up to roughly 4 km with a fixed precision of 1 µm, which should be enough for most use-cases.

* 1. **Considerations for timed metadata**

It may be desirable to also be able to store camera extrinsics as timed metadata tracks. If so, it would be useful if the same structure could be used for both the item property and timed metadata.

To efficiently do that, the following changes are needed:

1. A variable bit-precision is needed so that the struct size can be kept down.
2. The unit of pos\_x, pos\_y and pos\_z needs to be variable and not fixed to µm.
3. The denominator of quat\_x, quat\_y and quat\_z needs to be variable.
4. It needs to be possible to store predicted and absolute values.

A class like the following could be defined in 23090-7:

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) {  
 signed int((quat\_bytes\_minus1+1)\*8) quat\_x;  
 signed int((quat\_bytes\_minus1+1)\*8) quat\_y;  
 signed int((quat\_bytes\_minus1+1)\*8) quat\_z;  
 }  
};

Where:

**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 in the form denominator = 2quat\_den\_bits\_minus1+1. Valid values for quat\_den\_bits\_minus1 are in the range from [0, 13].

pos\_x, pos\_y, pos\_z, quat\_x, quat\_y**,** andquat\_z are otherwise specified as in 1.3.

With this class in place, the camera extrinsics item property box could then be defined as:

aligned(8) class CameraExtrinsicsMatrix  
extends ItemFullProperty('cmex', version = 0, flags) {  
 CameraExtrinsics extr\_matrix(1, flags & 0x7, 3, 0, 1, 13);  
 if (flags & 0x8) {  
 unsigned int(32) id;  
 }  
}

1. **Camera intrinsic matrix property box**

In order to make use of the multiple viewpoints in a multi-camera setup, the characteristics of each camera are also needed. This is typically specified via the camera intrinsics matrix.

One general form of specifying the intrinsics matrix for a pinhole camera is 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

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

* 1. **Definition**

Box Type: 'cmin'  
Container: ItemPropertyContainerBox (‘ipco’)  
Mandatory (per item): No  
Quantity (per item): Zero or one

The CameraIntrinsicsMatrix allows writers to communicate the characteristics of the camera that captured the associated image item.

* 1. **Syntax**

aligned(8) class CameraIntrinsicsMatrix  
extends ItemFullProperty('cmin', version = 0, flags) {  
 // denominator\_bits = (flags & 0x001F00) >> 8;  
 // skew\_denominator\_bits = (flags & 0x1F0000) >> 16  
 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;  
 }  
}

* 1. **Semantics**

**flags**: is a 24-bit integer with flags; the following values are defined:

intrinsics\_mode: Flag mask is 0x000001. The value 0 indicates that simplified intrinsics (no skew, square pixels) are used. The value 1 indicates that full intrinsics are used.

denominator: Flag mask is 0x001F00. The number of bits for the denominator is defined as (flags & 0x001F00) >> 8. The denominator itself is calculated as:  
denominator = 1 << denominator\_bits.

skew\_denominator: Flag mask is 0x1F0000. The number of bits for the skew denominator is (flags & 0x1F0000) >> 16. The denominator itself is calculated as:  
skew\_denominator = 1 << skew\_denominator\_bits.

focal\_length\_x**:** Specifies the horizontal focal length of the camera in image widths.

focal\_length\_y**:** When *intrinsics\_mode* is 1, specifies the vertical focal length of the camera in image heights. When not present (*intrinsics\_mode* is 0), the value shall be implied to be *focal\_length\_x \* image\_width / image\_height*.

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**:** Camera system skew factor. When not present its value shall be implied to be 0.

The values in the intrinsics matrix in section 2, can then 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 / skew\_denominator

Where image\_width and image\_height come from the 'ispe' associated with the image item.

* 1. **Discussion**

See 1.4 on the justification for keeping values as integers where possible.

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

The current proposal only deals with the pinhole camera model. Other models can be added in the future as a new version of this box.

1. **Alternatives considered**

There are several other specifications that define camera extrinsics and intrinsics. 23090-2 (OMAF), 23090-10 (V3C Carriage) and 23090-12 (MPEG Immersive Video) all contain relevant boxes. The camera extrinsics in this proposal matches the text in 23090-12 except for using integers instead of floating-point values.

For camera intrinsics, the other specifications tend to deal with viewports or fisheye lenses, neither of which maps very well to a more generic pinhole camera intrinsics matrix.

# Carriage of Text Items

*[ Ed. (FD):* Open question from MPEG#137 meeting: *“we need to find a reasonably manageable/compact language: text/plain says too little (well, nothing) about styling, and full-on HTML seems a bit much”].*

This topic was discussed as [HEIF Issue#66](http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/HEIF/-/issues/66) on the MPEG GitLab.

1. **Introduction**

Creating images with textual content such as captioning of images, memes and so on is very common and is done globally using various editing tools. Figure 1, shows an example meme image with plain text.



**Figure 1: Example of an meme image with plain text.**

The ISO/IEC 14496-30 standard on Timed text and other visual overlays in ISO base media file format specifies the carriage of timed text and subtitle streams in ISO BMFF tracks. However, neither ISO/IEC 14496-30 standard nor the HEIF standard specify the carriage of text items associated with an image item.

In MPEG 136, Nokia proposed a new item type called the text item for the carriage of textual content associated with an image item in m58143. The ISOBMFF group noted the proposal and provided the following suggestion:

We need to find a middle ground between plain text with no styling and the full complexity of something like HTML or SVG. We'd like to use but not overload the current technologies (such as overlay, MIME typed items, and so on).

Based on the above suggestions, this contribution proposes a mime type item for renderable text. The data in the mime type item is a renderable text.

1. **Proposal**

*Add following definitions in clause 3*

**3.1.X**

**text item**

*item* (3.1.27) whose data is the textual data.

**3.1.Y**

**renderable text item**

*a text item* (3.1.X) that includes possibly size, position, direction, language, font and styling and whose processing produces an output which can be visually rendered.

**3.1.Z**

**font item**

*item* (3.1.27) whose data is the fonts

*Update the subclause 6.5.2.1 as follows (changes highlighted in yellow)*

**6.5.2.1 Image spatial extents**

**6.5.2.2 Definition**

|  |  |
| --- | --- |
| Box type: | 'ispe' |
| Property type: | Descriptive item property |
| Container: | ItemPropertyContainerBox |
| Mandatory (per item): | Yes |
| Quantity (per item): | One |
|  |  |

The ImageSpatialExtentsProperty documents the width and height of the associated image item. Every image item shall be associated with one property of this type, prior to the association of all transformative properties.

The ImageSpatialExtentsProperty may be associated with items whose output can be visually rendered (e.g., renderable text items). When ImageSpatialExtentsProperty is associated with items whose output can be visually rendered, they document the visually rendered width and height of the data which is output from the associated item.

**6.5.2.3 Syntax**

aligned(8) class ImageSpatialExtentsProperty  
extends ItemFullProperty('ispe', version = 0, flags = 0) {  
 unsigned int(32) image\_width;  
 unsigned int(32) image\_height;  
}

**6.5.2.4 Semantics**

image\_width specifies the width of the reconstructed image in pixels, as specified in 6.3.

image\_height specifies the height of the reconstructed image in pixels, as specified in 6.3.

NOTE Item properties, such as decoder configuration or layer selection, can affect the reconstructed image. As a consequence, the width and height of the reconstructed image depend on the presence and content of such properties.

When ImageSpatialExtentsProperty is associated with items whose output can be visually rendered, the image\_width and image\_height specifies the visually rendered width and height, respectively of the data which is output from the associated item.

*Add the following subclause in Clause 6*

**6.A Text item and Renderable text item**

**6.A.1 Definition**

A text item is an item with item\_type value set to 'mime' and the data in the text item is text, for example, ‘html’ or ‘plain text’. The content\_type in ItemInfoEntry of the ItemInfoBox is set equal to the mime type of the data in the text item. Example values for content\_type field may include ‘text/html’ for html formatted text or ‘text/plain’ for plain text.

The text item is associated with the image item on which the textual data is displayed/rendered using an item reference of type 'cdsc' from the text item to the image item. A text item shall be associated with multiple image items only when all the associated image items have the same size. [NOTE: As an alternate to the use of the text item may be used as an overlay to the image item. However, this aspect needs to be further discussed]

The text item may be associated with the font item using an item reference of type 'font' from the text item to the font item. The font item carries the fonts used for rendering the text item.

The text item is associated with the ImageSpatialExtentsProperty which documents the visually rendered width and height of the data which is output from the text item.

The text item is associated with the TextLayoutProperty which documents the visually rendered size, position and language of the data which is output from the text item.

When a text item is not associated with any item or item property which documents possibly size, position, direction, language, font and styling for visual rendering, then, the data in the text item should contain the textual data together with possibly size, position, direction, language, font and styling for visual rendering of the text item and is called the renderable text item.

The renderable text data may be further encoded with either gzip or deflate or any other alogithm defined for content-encoding of Http/1.1. The encoding of renderable text data shall be defined by the content\_encoding parameter in ItemInfoEntry of the ItemInfoBox for the mime type text item.

If the renderable text data is encoded with any of the alogithm defined for content-encoding of Http/1.1, the data needs to be decoded before interpreting it as the mime type text item identified by the content\_type in ItemInfoEntry of the ItemInfoBox.

If the content\_encoding parameter in ItemInfoEntry of the ItemInfoBox has an empty string, then no content encoding is applied on the renderable text data.

The mime type item of renderable text only carries the text data required for rendering, however it does not provide any information on the display/layout conditions, for example the position, size and direction of the renderable text. Hence we propose a item property for the mime type text item which carries the information on the display/layout conditions.

[NOTE: As an alternate to the use of text layout property, the following approach may be used The ImageSpatialExtentsProperty to document the width and height,

The ImageOverlay to document the reference width and reference height and the position of the renderable text item

The ExtendedLanguageBox to document the language of the textual data. However, these aspect needs to be further discussed]

**6.B Text Layout Information**

**6.B.1 Definition**

Box type: 'txlo'

Property type: Descriptive item property

Container: ItemPropertyContainerBox

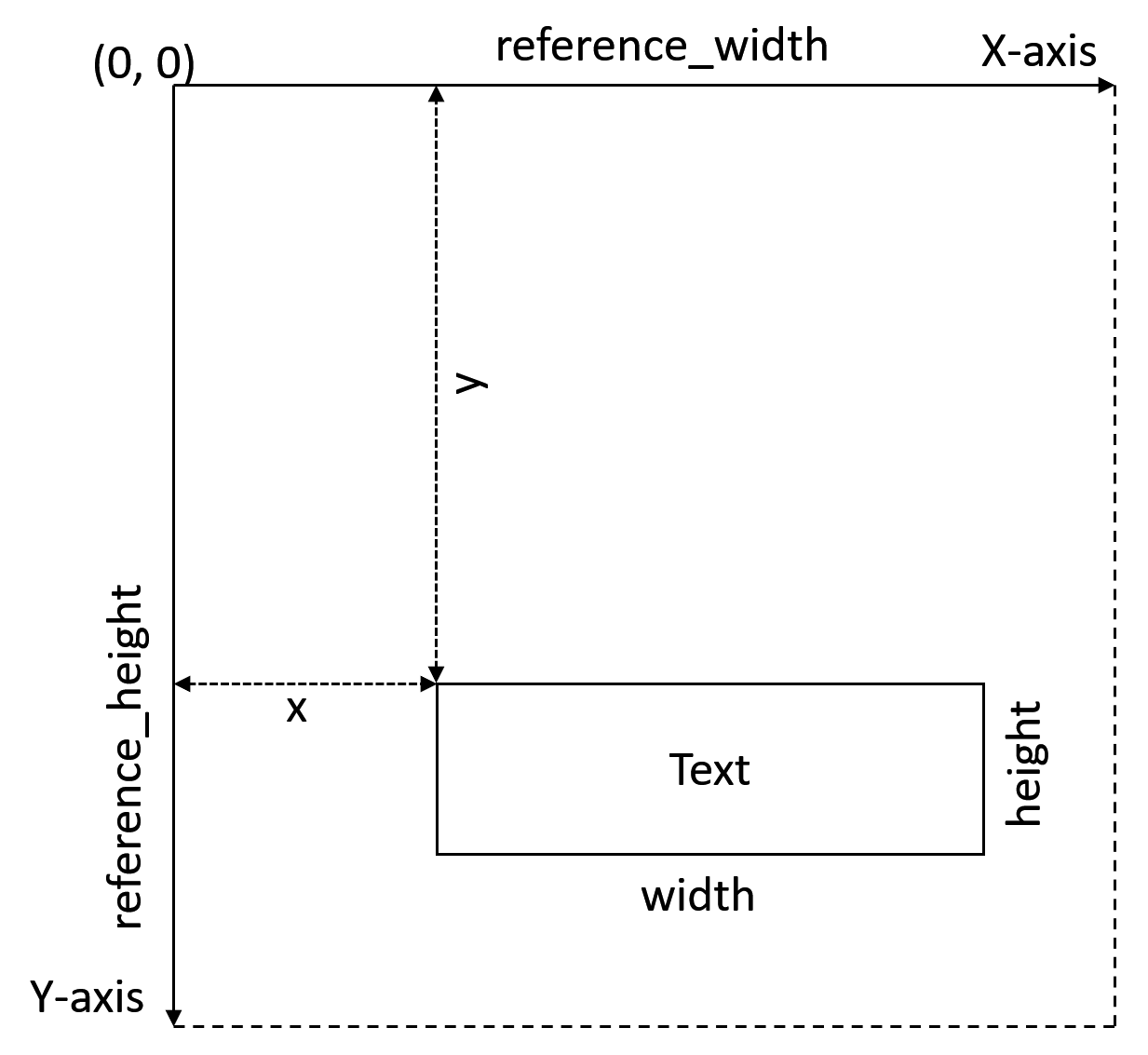
Mandatory (per item): No

Quantity (per item): One

The TextLayoutProperty documents the layout information of the associated text item. A text item shall be associated with one TextLayoutProperty prior to the association of all transformative properties.

The data in the TextLayoutProperty define the size, position and the language of the textual data to be displayed/rendered on the associated image item. The size and position information are used to display/render the textual data inside a reference space that is mapped to the image item with which the text item is associated after any transformative item property is applied to the image 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. Figure 2, provides an illustration of text item in the reference space. The placement of textual data inside the associated image item is obtained after applying the implicit resampling caused by the difference between the size of the reference space and the size of the associated image item. If the text item has transformative item properties, then the implicit resampling shall be performed on the text item before the first of its transformative item properties is applied.



**Figure 2: An illustration of text item in reference space.**

**6.B.2 Syntax**

aligned(8) class TextLayoutProperty  
extends ItemFullProperty('txlo', version = 0, flags = 0) {  
 unsigned int (8) version = 0;   
 unsigned int (8) flags;   
 field\_size = ((flags & 1) + 1) \* 16;   
 unsigned int(field\_size) reference\_width;   
 unsigned int(field\_size) reference\_height;  
 signed int(field\_size) x;  
 signed int(field\_size) y;   
 utf8string language;   
}

**6.B.3 Semantics**

version shall be equal to 0.

(flags & 1) equal to 0 specifies that the length of the fields x, y, width, height is 16 bits. (flags & 1) equal to 1 specifies that the length of the fields x, y, width, height 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 text items are placed.

x, y specify the top, left corner of the text item relatively to the reference space. The value (x = 0, y = 0) represents the position of the top-left pixel in the reference space.

NOTE Negative values for the x or y fields enable to specify top-left corners that are outside the image. This can be useful for updating text items during the edition of an HEIF file.

language is a character string containing an RFC 5646 compliant language tag string, such as "en-US", "fr-FR", or "zh-CN“, representing the language of the text. When language is empty, the language is unknown/undefined.

**6.C Font item**

**6.C.1 Definition**

A font item is an item with the item\_type value set to 'mime' and the data in the font item are fonts for example ‘woff’ (Web Open Font Format) or ‘ttf’ (true type font). The content\_type in ItemInfoEntry of the ItemInfoBox is set equal to the mime type of the data in the font item. Example values for content\_type field may include ‘font/ttf’ for true type fonts or ‘font/woff’ for web open font format fonts.

The font item may be associated with the text item using an item reference of type 'font' from the text item to the font item.

The font data may be further encoded with either gzip or deflate or any other alogithm defined for content-encoding of Http/1.1. The encoding of font data shall be defined by the content\_encoding parameter in ItemInfoEntry of the ItemInfoBox for the font item.

If the font is encoded with any of the alogithm defined for content-encoding of Http/1.1, the data needs to be decoded before interpreting it as the mime type font item identified by the content\_type in ItemInfoEntry of the ItemInfoBox.

If the content\_encoding parameter in ItemInfoEntry of the ItemInfoBox has an empty string, then no content encoding is applied on the font data.

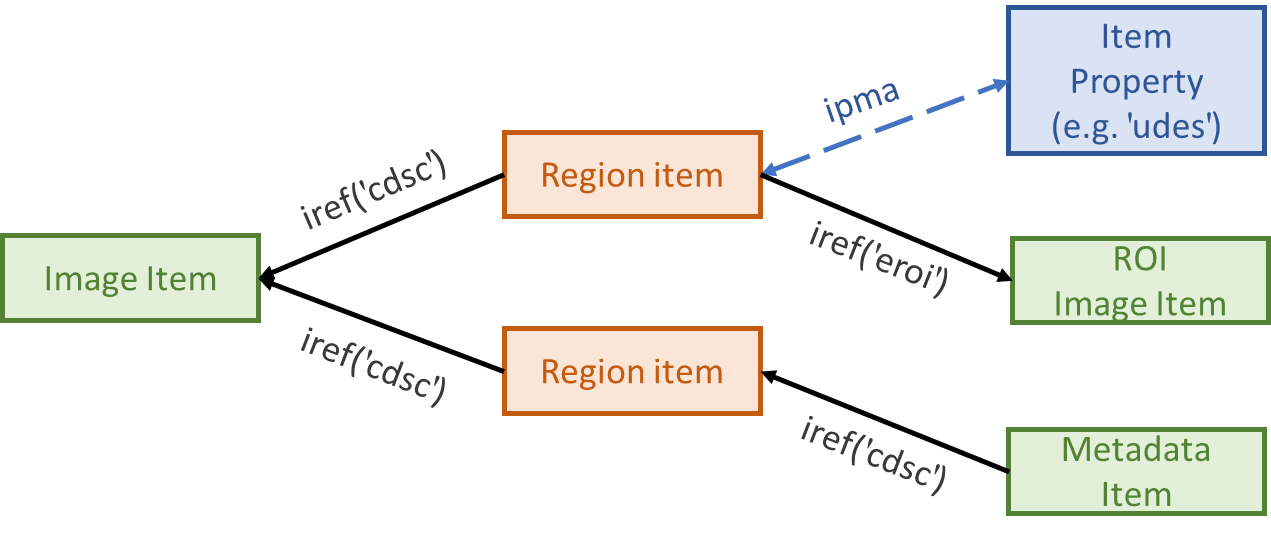
# Region annotations for image sequence or video tracks

This topic was discussed as [HEIF Issue#65](http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/HEIF/-/issues/65) on the MPEG GitLab.

1. **Motivation**

As illustrated in Figure 1, the HEIF 2nd edition specification [1] defines the association of a region annotation with one or more regions of an image item using the following constructions:

* describe in a region item the geometry of the one or more regions (where the geometry might have various shapes (polyline, ellipse, rectangle, mask, etc…));
* associate the region item with the image item it describes using a 'cdsc' item reference from the region item to the image item; and,
* associate any or all of the following with the region item for specifying annotations for this region item:
* descriptive image properties, using the ItemPropertyAssociationBox;
* 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 items or entity group.



**Figure 1: Example of association of region annotations with an image item**

Currently, HEIF doesn’t define an equivalent functionality for associating region annotations to samples of an image sequence or of a video track.

While one possibility may be to use a sample-to-item sample group ('stmi') to associate region items with runs of samples, the granularity of sample groups doesn’t allow an efficient description of regions as they have a high probability to be located at different locations in different samples.

We propose to define region annotations for tracks as follows:

* A region track with sample entries of type 'rgan', whose samples provide the geometry of one or more regions inside the samples of the associated image sequence or video track. The region track is associated with the image sequence or video track through a track reference of type 'cdsc'. This region track could be a ‘2dcc’ track as defined in MPEG-B Part-10, but a '2dcc' metadata track does not offer same possible geometries as the region annotation item.
* Annotation(s) associated with regions are specified using any of the following structures :
  + Item properties, possibly already associated with items, also associated with samples using a new sample-to-item-property sample group (‘stip’). For instance, this would be useful when an annotated item is also a sample in an image sequence;
  + Samples of a metadata track associated with the region track using a track reference 'cdsc' from the metadata track to the region track;
  + Samples of an image sequence or video track associated with the region track using a track reference 'eroi' from the region track to the image sequence or video track.

In the following section, we propose text to be integrated into ISOBMFF for defining the new sample-to-item-property sample group ('stip').

1. **Proposal**

*Define a sample-to-item-property sample group ('stip')*

**Y.Y Sample-to-item-property sample group**

**Y.Y.1 Definition**

Samples of a track can be linked to one more item properties using the sample-to-item-property sample grouping. The MetaBox containing the referred items is resolved as specified in the semantics section.

The sample-to-item-property sample grouping is allowed for any types of tracks, and its syntax and semantics are unchanged regardless of the track handler type.

**Y.Y.2 Syntax**

class SampleToItemPropertyEntry()

extends SampleGroupDescriptionEntry('stip') {

unsigned int(32) meta\_box\_handler\_type;

unsigned int(32) num\_properties;

for(i = 0; i < num\_properties; i++) {

unsigned int(32) property\_index[i];

}

}

**Y.Y.3 Semantics**

meta\_box\_handler\_type informs about the type of metadata schema used by the MetaBox which contains the item properties listed in this sample group. When there are multiple MetaBoxes with the same handler types, the MetaBox referred to in this sample group entry is the first MetaBox fulfilling one of the following ordered constraints:

- A MetaBox included in the current track, with handler\_type equal to meta\_box\_handler\_type.

- A MetaBox included in MovieBox, with handler\_type equal to meta\_box\_handler\_type.

- A MetaBox included in the root level of the file, with handler\_type equal to meta\_box\_handler\_type.

num\_properties counts the number of item properties referenced by this sample group.

property\_index[i] specifies the 1-based index (counting all boxes, including FreeSpace boxes) of an item property box, in the ItemPropertyContainerBox contained in the ItemPropertiesBox, that applies to or is valid for the samples mapped to this sample group description entry.

1. **Discussion**

At MPEG#137 meeting, it was commented that:

* It may be a better design to define region annotations within a sample group which is part of the sample group description entry of the track.
* The proposed approach (‘stip’) may be vulnerable to file editing.
  + For example, if item properties are reordered in editing, the editor should also check all tracks for this sample group and potentially modify the sample group description entries according to the reordered item properties
  + Also, If a MetaBox is added at a level that is higher in the precedence order than the originally referenced MetaBox (having the same handler type), the associations to item properties are broken, i.e. this sample group would reference a wrong MetaBox *(N.B.: same comment may apply to existing ‘stmi’ sample group in ISOBMFF)*
* an alternative would be a property sample group

The following questions are to be considered:

* do ‘rgan’ tracks need a formal way of saying "the region in question doesn't appear over this time range"?
* may property of a region change over time?
* would it be more robust to associate samples with items, and say that the samples inherit the properties of the items? It's both more compact, and leaves property association in the item properties box.…

# Matrix-based transformation for image items

*[[ Ed. (FD): MPEG#129: it was questioned:”* Should we also add ‘matrix’ as an image derivation in the HEIF? “. It was warned that “We would need to be clear about the meaning of outputs that don’t have horizontal and vertical sides; if that’s overlaid, the meaning is clear, but what if it’s supposed to be displayed?”*]]*

# Signaling for pre-derived coded image items

*Replace the clause 6.4.7 with the following text:*

**6.4.7** **Pre-derived coded images**

[Ed. (FD): In the following, differences with HEIF 2nd edition (w18310) are highlighted in blue]

If a coded image has been derived from others — for example, a composite HDR image derived from exposure-bracketed individual images, or a panorama derived from a set of images — then it shall be linked to those images by item references of type 'base'. Item references may be from the coded image to all images it derives from, or when unique IDs are used, from the coded image to all entity groups or images it derives from. When unique IDs are used, a to\_item\_ID value in the SingleItemTypeReferenceBox or SingleItemTypeReferenceBoxLarge is resolved to an item identifier whenever the embedding MetaBox contains an item with such identifier, and is resolved to an entity group identifier otherwise.

An image item including a 'base' item reference is referred to as a pre-derived coded image.

NOTE In this version of this document, the exact derivation process used to produce the image is not described.

[[Ed. (FD): At MPEG#129, it was commented that “The slight snag here is defining what it means when the entity group does NOT imply a single output (e.g. a slide show); what does pre-derivation mean? ]]

*Add the following clause as section 6.4.7.1:*

**6.4.7.1 Signaling of the derivation method for pre-derived coded image items**

A pre-derived coded image shall be linked to images it derives from by an item reference of type 'base' to the entity group containing all images the pre-derived coded images derives from. The grouping\_type of the EntityToGroupBox specifies the purpose of grouping and implicitly signals the type of the derivation operation which was applied to generate the pre-derived coded image.

[[Ed. (FM): At MPEG#126, it was commented that “we somehow need to indicate the derivation operation, rather than the nature of the input set”]]

[[Ed. (FD): At MPEG#129, it was commented that “We could allow a pre-derivation of the implied derivation of that entity group.”]]