ISO/IEC JTC 1/SC 29/WG 03 N0332

**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:** 2021-07-16

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

**No. of pages:** 9 (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 N0332**

**July 2021, Virtual**

|  |  |
| --- | --- |
| **Title** | **Technology under Consideration on ISO/IEC 23008-12** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **20586** |

# Abstract

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

Contents

[Abstract 1](#_Toc77351792)

[1. On progressive decoding 1](#_Toc77351793)

[1. Context 2](#_Toc77351794)

[2. Discussions at MPEG#133 2](#_Toc77351795)

[3. Update at MPEG#134 3](#_Toc77351796)

[Progressive Rendering Algorithm 3](#_Toc77351797)

[Progressive Entity Group 4](#_Toc77351798)

[Progressive Layer Information Item Property 4](#_Toc77351799)

[Progressive Derived Image Item Information Item Property 5](#_Toc77351800)

[2. Matrix-based transformation for image items 7](#_Toc77351801)

[3. Signaling for pre-derived coded image items 7](#_Toc77351802)

[4. Possible optimization for region annotations 8](#_Toc77351803)

[Comparative analysis of storing 10 annotation regions in an image 8](#_Toc77351804)

# On progressive decoding

Discussion available under:

<http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/HEIF/-/issues/33> (MPEG#134)

<http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/HEIF/-/issues/55> (MPEG#135)

Contribution m56028 is a continuation of the discussions started in MIAF at MPEG#132 (m55218) regarding support of progressive decoding and discusses progressive refinement for grid items.

## Context

At the MPEG 132nd meeting, the contribution “*On Progressive Decoding for MIAF*” (m55218) recalled that one possible usage of the ‘altr’ entity group is to group together several items for progressive decoding. That contribution proposed (pasted below for convenience) to define a brand to signal that this is the intended use of the ‘altr’ in an HEIF file. That contribution stated in particular:

*Progressive decoding/refinement can be achieved by grouping together several items in an ‘altr’ group or by using thumbnails. If the highest quality/resolution item is placed first in the group but the item data is last in the file, a decoder may potentially decode and display the other items in the group first until it has downloaded all of the data.*

*There is, however, currently not any way of telling a parser/decoder that the purpose of the ‘altr’ group is for progressive refinement. We need some way, like a brand, for the encoder to signal that this is the intended use of the ‘altr’ group.*

While this proposal provides a sufficient description for “simple” image items, it is not always sufficient for derived items like grid or overlay items. Indeed, progressive decoding for a grid (or overlay) derived image item may be realized by providing alternative image items for each input image of a grid (or overlay) derived image item.

At MPEG 135, contribution m57385 proposed examples and use cases for progressive rendering for derived image items based on the results of MPEG#134 (‘prgr’ group and ‘prdi’ or ‘prli’ image properties) to specify an ordering of image data. Discussions during the meeting led to the discussion of possible alternatives to ‘prdi’ or ‘prli’ that have been put under consideration.

## Discussions at MPEG#133

A better approach to the use of 'altr' may be to create a dedicated group for progressive rendering ('prgr'?) and to recommend or mandate:

* a 'prgr' entity group shall only list items, not tracks
* the items within 'prgr' shall be listed in order from the smallest resolution to the largest
* it is recommended that the item\_data needed to decode them be stored in order also, such that a client progressively receiving a file can perform progressive display as item data becomes available.
* indication of when to meaningfully produce an updated rendering (for progressive-download aware player)

For the last bullet, a property associated to a grid (or overlay) item was proposed:

aligned(8) class ProgressiveStepRefinementProperty  
extends ItemFullProperty('prco', version = 0, flags = 0) {  
 unsigned int(16) step\_count;   
 for (i=0; i < step\_count; i++) {  
 unsigned int(16) item\_count;   
 for (j=0; j < item\_count; j++) {  
 if (flags & 1) {  
 unsigned int(32) item\_index;   
 }  
 else {  
 unsigned int(16) item\_index;   
 }  
 unsigned int(16) extent\_count;   
 for (k=0; k < extent\_count; k++) {  
 unsigned int(16) extent\_index;   
 }  
 }  
 }  
}

Where

step\_count indicates the number of progressive steps described inside the progressive step refinement item property.

item\_count value indicates how many items are comprised in the progressive step’s refinements.

item\_index value indicates the index of a resource inside the array in the item location box, 'iloc'.

## Update at MPEG#134

The input contribution m56773 proposes to use three complementary mechanisms for describing the progressive rendering of an item, with a specific handling of layer-based items and derived image items.

The discussion is available under:

<http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/HEIF/-/issues/48>

**Proposal**

Three mechanisms can be used to describe the organization of an HEIF file suitable for progressive rendering:

* The ‘prgr’ entity group enables to group several items in increasing quality order that can be used to perform a progressive display of one of the latest items of the group.
* The ‘prli’ item property signals layers of an image item that can be used for a progressive display of the associated image item.
* The ‘prdi’ item property specifies progressive rendering steps for an associated derived image item using variable-quality versions of its input image items.

In addition, a specific brand could be used to indicate that the content of an HEIF file is organized for allowing progressive rendering.

### Progressive Rendering Algorithm

The progressive rendering of a target item from an HEIF file can be realized with the following algorithm:

1. If the target item is contained in a ‘prgr’ entity group, then any item occurring before it inside this ‘prgr’ entity group may be rendered as a temporary replacement of the target item.
2. If the target item is associated with a ‘prli’ item property, then any set of layers signaled by the ‘prli’ item property may be used for a low-quality and temporary rendering of the target item.
3. If the target item is associated with a ‘prdi’ item property, then any progressive rendering step described by the ‘prdi’ item property may be used for a low-quality and temporary rendering of the target item.
4. If the target item is a derived image item with a single input image and no associated ‘prdi’ item property and its input image supports progressive rendering, then the derived image item image may be reconstructed from any progressive rendering of its input image.

An item can be contained in a ‘prgr’ entity group and also be associated either with a ‘prli’ item property or a ‘prdi’ item property.

An item shall not be associated both with a ‘prli’ item property and a ‘prdi’ item property.

A derived image item with several input images supporting progressive rendering should have an associated ‘prdi’ item property to indicate a limited set of progressive rendering steps for the derived image item based on the input images progressive rendering.

### Progressive Entity Group

The ‘prgr’ entity group signals a set of items that can be used for a progressive rendering of one of these items.

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

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

A ‘prgr’ entity group shall only contain items, not tracks.

At MPEG#135, there was a consensus on this proposal. However, a NOTE may be added to recommend that image items of the same 'prgr' entity group should be members of the same 'altr' entity group so that legacy players could treat the image items as alternatives and display one of them (which is better than displaying them some other way like a collection).

### Progressive Layer Information Item Property

The progressive layer information item property, ‘prli’, describes progressive rendering steps associated to an item encoded with layers. Each progressive rendering step indicates the encoding layers used for rendering the image item at this step.

Each progressive rendering step is described as a difference from the previous step. This difference specifies the byte range inside the item’s data corresponding to the layer or layers added by the progressive rendering step.

Note: several encoding layers may be grouped into a single progressive rendering step.

Syntax:

aligned(8) class ProgressiveLayerInformationProperty

extends ItemFullProperty('prli', version = 0, flags) {

field\_length = ((flags & 1) + 1) \* 16;

unsigned int(8) step\_count;

for (i=0; i < step\_count; i++) {

unsigned int(field\_length) step\_size;

}

}

Semantics:

* field\_length: the size in bits of the layer\_size field. Either 16 or 32.
* step\_count: the number of progressive steps based on layers described in the item property.
* step\_size: the size in bytes of data corresponding to the progressive step being described. This size corresponds to one or more layers of the image item to which the item property is associated.

Editor Note: for optimization purpose, the size for the last progressive step could be omitted or set to 0.

#### Alternative to ‘prli’ suggested at MPEG#135

It has been suggested the use of Sub-sample item property ‘subs’ (see [this comment on MPEG GitLab](http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/HEIF/-/issues/55" \l "note_43501)) to indicate the layer sizes in bytes, using picture-based sub-samples (flags == 5), since ‘subs’ is already available in HEIF. It was commented that

* this flags value is not defined for all codecs (AVC for example) and
* using 'subs' property makes the parsing codec-dependent. From one codec to another ‘subs’ doesn't have the same fields, for example in VVC, the layerID field is not available (while present in HEVC) and
* ‘subs’ property is limited to NALU-based codecs and would make HEIF less generic and extensible

An issue in the use of ‘prli’ was pointed out in this [other GitLab comment](http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/HEIF/-/issues/55" \l "note_43665): The usage of 'prli' would require the decoder to output intermediate frames, which is functionality beyond standard-conforming operation. Indeed, a standard-conforming HEVC or VVC decoder that decodes a quality or spatial scalable bitstream would output the highest layer of the bitstream only, at least that would be typically the way how the output layer sets are indicated in the bitstream. It was argued that using ‘subs’ would suit standard-conforming decoders, would achieve achieve the same functionality as 'prli', and would not require specifying new HEIF structures.

We then have **an open question for progressive rendering using layered images**: ‘prli’ versus ‘subs’ approach.

### Progressive Derived Image Item Information Item Property

The progressive derived image item information item property describes progressive rendering steps associated with a derived image item. Each progressive rendering step specifies which replacement image to use in place of each input image for the reconstruction of the derived image item. A replacement image may either be an empty image, a lower-quality version of the input image, or the input image itself. When reconstructing the derived image item for a progressive rendering step, an HEIF viewer may handle any empty replacement image as best adapted, for example using a blank image or a transparent image.

Each progressive rendering step is described as a difference from the previous step. This description lists replacement images to be used to reconstruct the derived image item. Initially, before the first progressive rendering step, replacement images for input images all correspond to empty images. Each progressive rendering step adds new images as replacement images and/or updates existing replacement images with other replacement images.

The progressive derived image item information item property can describe two different cases of replacement images. First, an input image can be replaced by a lower quality image associated to it through a ‘prgr’ entity group. Second, an input image can be replaced by a lower quality version corresponding to a reduced set of layers composing this input image and described through a ‘prli’ item property. It can also handle the case where an input image is both contained in a ‘prgr’ entity group and has an associated ‘prli’ item property.

Note: the ‘prdi’ item property is intended to be used with derived image items using several input images.

A derived image item with a single input image and no associated ‘prdi’ item property may be rendered progressively by using the progressive rendering information associated to its input image.

When a derived image item with a single input image and an associated ‘prdi’ item property is rendered progressively, it shall be rendered progressively according to the information contained in the ‘prdi’ item property.

When a derived image item has several input images and some of its input images can be rendered progressively, the derived image item should have an associated ‘prdi’ item property. If the derived image item has no associated ‘prdi’ item property, the HEIF viewer may render it progressively as best adapted to its needs.

Syntax:

aligned(8) class ProgressiveDerivedImageItemInformationProperty

extends ItemFullProperty('prdi', version = 0, flags = 0) {

unsigned int(8) step\_count;

for (i=0; i < step\_count; i++) {

unsigned int(16) item\_count;

for (j=0; j < item\_count; j++) {

unsigned int(16) item\_index;

unsigned int(8) step\_index;

unsigned int(8) layer\_index;

}

}

}

Semantics:

* step\_count: number of progressive steps associated to the derived image item reconstruction.
* item\_count: number of progressive input image replacements updated by the progressive step.
* item\_index: index of an input image updated by the progressive step. This index is a 0-based index inside the ‘dimg’ item reference of the derived image item.
* step\_index: if the image item identified by the item\_index field is contained in a ‘prgr’ entity group, the step\_index field indicates the 0-based index of the item inside this ‘prgr’ entity group to use as the replacement image for this input image for the progressive step being described. Otherwise, the step\_index field is set to the value 0.
* layer\_index: if the image item identified by the item\_index field has an associated ‘prli’ item property, the layer\_index field indicates the 0-based index of the layer-based progressive step inside this ‘prli’ item property to use as the replacement image for this input image for the progressive step being described. Otherwise, the layer\_index field is set to the value 0.

Note: if the input image is neither contained in a ‘prgr’ entity group nor associated with a ‘prli’ item property, then both the step\_index and layer\_index shall have the value 0, and the input image itself is used for reconstructing the derived image item.

Note: if the input image is both contained in a ‘prgr’ entity group and associated with a ‘prli’ item property, the information inside the ‘prli’ item property shall be taken into account only if the step\_index corresponds to the position of the input image inside the ‘prgr’ entity group. Otherwise, the layer\_index shall be set to 0 and the information inside the ‘prli’ item property shall be ignored.

#### Alternative to ‘prdi’ suggested at MPEG#135

It has been suggested (see [this comment on MPEG GitLab](http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/HEIF/-/issues/55" \l "note_43501)) that the functionality offered by ‘prdi’ could be achieved by other means already available in HEIF as follows:

* Create an overlay image item per each "step index". Use a scaled thumbnail image item as the "background image" in the overlay image item and the "high-quality" image items in this step index as the other input images for the overlay.
* Include the grid and overlay image items in the same 'prgr' entity group.

It was commented that this approach would put more burden on the writer to write all intermediate items and to take care about data interleaving/order between input images and for example grid or overlay data. It would also put more burden on the player, the list of intermediate items being seen as multiple independent reconstructions whereas with 'prdi' indication, players would progressively reconstruct a same item; i.e. parts of the grid may not require update from intermediate one step to another.

We then have this **open question regarding progressive rendering for derived images**:

* Should the progressive steps being defined precisely (e.g. items and transformation operations)?, or
* Should the player deduce what is possible to render?
  + In this case, could information like 'prdi' help the player?

# 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 overlayed, 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.”]]

# Possible optimization for region annotations

Discussion available under:

<http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/HEIF/issues/20>

Revised text for HEIF CDAM3 contains a dedicated item definition for describing regions using a single construct (square, circle, etc…). At 1st WG03 meeting, m55123 proposed the use of an array of constructs to enable efficient storage of regions which may be associated with the same annotation (e.g. face, person, car, pet, etc.). It would also enable “instantiation” by only listing one region in the array (e.g. “Bob”, “My car”, “my dog”, etc.).

Question was raised on a potential conflict on cases where a region is

* a group of single constructs where the region is the union of the geometries (e.g. you 'frame' a complex shape by laying a lot of rectangles over it)
* a group, where the statements about the region apply to each geometry independently.

## Comparative analysis of storing 10 annotation regions in an image

The following table provides a comparative analysis of storing 10 regions which belong to faces in an image.

|  |  |
| --- | --- |
| **Storage as deductive information (m55123)** | **Storage with a ‘rgan’ item (revised CDAM3)** |
| 1 URI item (with an array of 10 region data structures)  1 iloc entry  1 iinf entry  1 iref 'cdsc' (to image item)  1 UUID item property (indicating that stored information are faces)  1 ipco entry  1 ipma entry  (‘dpnd’ not needed in this example) | 10 'rgan' items (with one region data each)  10 iloc entries  10 iinf entries  10 iref 'cdsc' entries (from region item to the image item)  1 UserDescriptionProperty (for tagging that the annotations are faces)  1 ipco entry  1 ipma entry |
|  | **Conclusion: 9 additional items, iloc and iref entries stored (total of 27 additional entries)** |