ISO/IEC JTC 1/SC 29/WG 03 N1299

**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:** 2024-08-02

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

**No. of pages:** 17 (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 N1299**

**July 2024 – Sapporo, Japan**

|  |  |
| --- | --- |
| **Title** | **Technology under Consideration on ISO/IEC 23008-12** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **24145** |

Abstract

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

Table of Contents

[1 Region annotations for image sequence or video tracks 3](#_Toc173271705)

[1.1 Region extrapolation (from m60304, MPEG#139, Issue#76) 3](#_Toc173271706)

[1.1.1 Overview 3](#_Toc173271707)

[1.1.2 Text Proposal 3](#_Toc173271708)

[1.2 Region interpolation (from m59508, MPEG#138, Issue#69 comment#60556) 6](#_Toc173271709)

[1.2.1 Text Proposal 6](#_Toc173271710)

[1.2.2 Discussion 6](#_Toc173271711)

[2 Region annotation for image items (from m62028, MPEG#141, Issue#88 and MPEG #145, issue #128) 7](#_Toc173271712)

[2.1 Discussion 7](#_Toc173271713)

*[2.1.1](#_Toc173271714)**[Motivation, use cases and initial proposal](#_Toc173271714)* [7](#_Toc173271714)

*[2.1.2](#_Toc173271715)**[Responses to open questions](#_Toc173271715)* [9](#_Toc173271715)

[2.2 Proposal 10](#_Toc173271716)

[2.2.1 Proposal 1: Union of regions 10](#_Toc173271717)

[11.3.3.2.2 Union derivation 10](#_Toc173271718)

[2.2.2 Proposal 2: Relations between region items 11](#_Toc173271719)

[11.3.4 Region Entity Group 11](#_Toc173271720)

[11.3.4.1 ‘corg’ Entity Group 11](#_Toc173271721)

[3 Matrix-based transformation for image items 11](#_Toc173271722)

[4 Signaling for pre-derived coded image items 11](#_Toc173271723)

[5 On MPEG/JPEG file embedding (MPEG#141, Issue#87) 12](#_Toc173271724)

[5.1 Discussion 12](#_Toc173271725)

[5.2 Initial text proposal 12](#_Toc173271726)

[6 Disparity adjustment property for frame-packed stereo pair (MPEG #144, issue #111) 12](#_Toc173271727)

[7 SlimHEIF design with Compressed MetaBox (MPEG #146, issue #151)) 13](#_Toc173271728)

[7.1 Abstract 13](#_Toc173271729)

[7.2 Proposal 13](#_Toc173271730)

[7.3 References 13](#_Toc173271731)

[8 Increasing Grid Rows and Columns in HEIF (MPEG #147, issue #160)) 13](#_Toc173271732)

[8.1 Abstract 13](#_Toc173271733)

[8.2 Use case details 13](#_Toc173271734)

[8.3 Requirements 14](#_Toc173271735)

[8.4 Alternative Approaches 15](#_Toc173271736)

[8.4.1 Approach #1: 15](#_Toc173271737)

[8.4.2 Approach #2: 15](#_Toc173271738)

[8.4.3 Approach #3: 16](#_Toc173271739)

[8.4.4 Approach #4: 16](#_Toc173271740)

[9 Generic compression of items (MPEG #147, issue #157)) 16](#_Toc173271741)

[9.1 Abstract 16](#_Toc173271742)

# Region annotations for image sequence or video tracks

## Region extrapolation (from [m60304](https://dms.mpeg.expert/doc_end_user/documents/139_OnLine/wg11/m60304-v1-m60304-Regionextrapolationfortracks.zip), MPEG#139, [Issue#76](https://mpeg.expert/software/MPEG/Systems/FileFormat/HEIF/-/issues/76))

### Overview

A black background with white squares

Description automatically generated

Figure 1: example region description for tracks using extrapolation

Figure 1 shows an example of describing several regions using extrapolation. The video track shown at the top contains two regions, an elliptic one and a rectangular one. The elliptic one is present in the four first samples of the video track and moves to the right of the image. The rectangular one is present in all the samples of the video track and moves to the left of the image.

The region track shown at the bottom describes these two regions. In a first sample, corresponding to the first sample of the video track, these two regions are described with their positions and sizes and the evolution of their respective positions and sizes. There are no region samples corresponding to the three following video samples. The region sample corresponding to the fifth video sample signals that the interpolation of the elliptic region ends.

### Text Proposal

*Update the definition of a region track ( section 7.5.4.1) by adding the following paragraphs:*

The geometry of a region may be defined by specifying the shape, position and size of the region in a sample of the region track. The geometry of a region may also be defined as an initial geometry and its evolution over time by specifying the initial geometry of the region and its evolution in a sample of the region track.

The evolution of a region over time is optional. It can be represented by the evolution speed of some of its parameters inside the reference space. The evolution speed of the parameters is signaled using a scaling factor for increasing its precision. The parameters defining the evolution of a region depend on the geometry of the region as follows:

— When the geometry of a region is represented by a point, the evolution of the region is defined by the evolution of the position of this point.

— When the geometry of a region is represented by a rectangle or an ellipse, the evolution of the region is defined by the evolution of the position and the size of the rectangle or ellipse.

— When the geometry of a region is represented by a polygon or a polyline, the evolution of the region is defined by the evolution of the position of each point of the polygon or polyline. The number of points in the polygon or polyline doesn’t change.

— When the geometry of a region is represented by a mask, the evolution of the region is defined by the evolution of the position of the mask.

The evolution of a region stops when another sample contains a region with the same region identifier. The evolution of a region shall stop for each sync sample of the source track.

*Update the Sample format (section 7.5.4.2.1) with the following paragraph*

When the extrapolate flag is set to 1 for a region inside a sample of a region track, the region is an evolving region defined by an initial geometry and its evolution over time.

The value of each evolving parameter defining the geometry of the region at a given composition time *T* can be computed as follows:

where:

* *param0* is the initial value of the parameter as defined in the initial geometry of the region at time T0.
* *Δparam* is the evolution of the parameter as defined in the evolution of the region.
* *evolution\_scale* is a scaling factor for the evolution values equal to , where is the field\_size and is equal to ((RegionTrackConfigBox.field\_length\_size & 1) + 1) \* 16.
* *T0* is the composition time of the sample defining the evolving region.
* *ΔT* is the duration of the sample defining the evolving region.

*Update the syntax of Sample format (section 7.5.4.2.2) as follows*

aligned (8) class RegionSample {  
 unsigned int field\_size = ((RegionTrackConfigBox.field\_length\_size & 1) + 1) \* 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;  
 unsigned int(8) geometry\_type;  
 unsigned int(1) extrapolate;  
 unsigned int(7) reserved;  
 if (geometry\_type == 0) {  
 // point  
 signed int(field\_size) x;  
 signed int(field\_size) y;  
 if (extrapolate == 1) {  
 signed int(field\_size) delta\_x;  
 signed int(field\_size) delta\_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;  
 if (extrapolate == 1) {  
 signed int(field\_size) delta\_x;  
 signed int(field\_size) delta\_y;  
 signed int(field\_size) delta\_width;  
 signed int(field\_size) delta\_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;  
 if (extrapolate == 1) {  
 signed int(field\_size) delta\_x;  
 signed int(field\_size) delta\_y;  
 signed int(field\_size) delta\_radius\_x;  
 signed int(field\_size) delta\_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;  
 }  
 if (extrapolate == 1) {  
 for (i=0; i < point\_count; i++) {  
 signed int(field\_size) delta\_px;  
 signed int(field\_size) delta\_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;  
 unsigned int(field\_size) track\_mask\_idx;  
 if (extrapolate == 1) {  
 signed int(field\_size) delta\_x;  
 signed int(field\_size) delta\_y;  
 }  
 }  
 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[];  
 if (extrapolate == 1) {  
 signed int(field\_size) delta\_x;  
 signed int(field\_size) delta\_y;  
 }  
 }  
 else if (geometry\_type == 7) {  
 // empty region  
 }  
 }  
}

*Update the semantics of Sample format (section 7.5.4.2.3) with the following text:*

7: the region is an empty region used for signalling the end of the evolution of a previous region with the same region identifier.

Other values are reserved.

extrapolate is a flag indicating whether the geometry changes of the region are specified or not. When equal to 0, it indicates that no geometry changes are specified for the region. When equal to 1, it indicates that both the geometry and the geometry changes are specified for the region.

(…)

evolution\_scale is the scaling factor for the specification of the evolution values, equal to , where is the field\_size and is equal to ((RegionTrackConfigBox.field\_length\_size & 1) + 1) \* 16.

delta\_x, delta\_y specify, in 1/evolution\_scale units of the reference space, the evolution of the x and y fields for the region.

delta\_width, delta\_height specify, in 1/evolution\_scale units of the reference space the evolution of the width and height fields for the region.

delta\_radius\_x, delta\_radius\_y specify, in 1/evolution\_scale units of the reference space the evolution of the radius\_x and radius\_y fields for the region.

delta\_px, delta\_py specify, in 1/evolution\_scale units of the reference space the evolution of the px, py fields for a point of the region.

## Region interpolation (from [m59508](https://dms.mpeg.expert/doc_end_user/documents/138_OnLine/wg11/m59508-v1-m59508-Regionannotationfortracks.zip), MPEG#138, [Issue#69 comment#60556](https://mpeg.expert/software/MPEG/Systems/FileFormat/HEIF/-/issues/69#note_60556))

*[Ed. (FD)This section only contains parts of the contribution requiring further discussion, i.e., the interpolate flag in sample format for region tracks]*

### Text Proposal

**X.X Region track and region annotations for an image sequence or video track**

**X.X.3 Sample format**

**X.X.3.1 Definition**

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

**X.X.3.2 Syntax**

aligned (8) class RegionSample {  
 unsigned int field\_size = ((RegionTrackConfigBox.field\_length\_size & 1) + 1) \* 16;   
// this is a temporary, non-parsable variable  
 unsigned int(7)reserved;  
 unsigned int(1)interpolate;  
 unsigned int(16) region\_count;  
 for (r=0; r < region\_count; r++) {  
 (…)  
 }  
}

**X.X.3.3 Semantics**

interpolate indicates the continuity in time of the successive samples. When true, the application may linearly interpolate values of the region geometries between the previous sample and the current sample. When false, there shall not be any interpolation of values between the previous and the current samples.

NOTE 1 When using interpolation, it is expected that the interpolated samples match the presentation time of the samples in the referenced source track. For instance, for each video sample of a video track, one interpolated region sample is calculated.

(…)

### Discussion

About the interpolate flag: The purpose is to avoid declaring a sample in the region track for each sample of the media track when regions are moving linearly between two positions. Imagine a sample A in the region track with a region at a starting position A and this region is moving linearly to the arrival position B nine samples later. Instead of declaring ten samples in the region track, you can only declare two samples, sample A with a duration corresponding to nine samples in the media track, followed by sample B providing the arrival position B. We should clarify that since the interpolate flag applies to all regions in the sample, the number of regions shall be the same in sample A and B.

# Region annotation for image items (from [m62028](https://dms.mpeg.expert/doc_end_user/documents/141_OnLine/wg11/m62028-v1-m62028-Regioncombination.zip), MPEG#141, [Issue#88](https://mpeg.expert/software/MPEG/Systems/FileFormat/HEIF/-/issues/88) and MPEG #145, issue [#128](https://git.mpeg.expert/MPEG/Systems/FileFormat/HEIF/-/issues/128))

*[[ Ed. (LB): MPEG#145: How do we unify the two approaches? (Entity group and derived region item)]]*

## Discussion

### *Motivation, use cases and initial proposal*

The HEIF 2nd edition specification [1] enables associating an annotation with a region of an image by defining a region item associated with an image item. Several regions of an image can be described either within a same region item (typically when the regions share the same annotations) or within different region items (typically when the regions do not share the same annotations).

Current specification does not provide specific tools for combining multiple annotated regions and annotating the corresponding union as a whole, or for signaling the aggregation of several regions forming logical parts of a same ‘object’. When an image is annotated, such tools are useful to document in the file format explicit relationships between regions rather than possibly inferring them at player-side.

The Figure 1 illustrates the former use case, when there are several regions inside an image, for instance, each corresponding to a different person, it is possible to associate the name of each person with the corresponding region by defining multiple region items (illustrated by identifiers ranging from 1 to 7) and their respective ‘label’ annotations, for instance using a UserDescriptionProperty. However, it is not possible to easily combine those annotated regions corresponding to different persons and to associate an annotation applying to the union as a whole, for instance indicating that they belong to the same family and possibly providing overall information on the family.

A black and white drawing of a group of people

Description automatically generated

Figure 1: Example of a region defined as a union of annotated regions

To address this former use case, a new type of derived region item representing the union of several regions defined as inputs is proposed.

The Figure 2 illustrates the second use case where a content creator wants to indicate that an ‘object’ in an image represented by a region is also a logical aggregation of several other regions of the image. The aggregation does not imply that the area of the region corresponding to the ‘object’ is the exact combination of areas of other aggregated regions. In Figure 2, a first region (in green) corresponds to a person and several other regions correspond to the head, body, arms and legs of this person respectively. However, it is not currently possible to indicate the relationship between the head, body, arm and leg regions and the region representing the whole body.

A screenshot of a computer

Description automatically generated

Figure 2: Example of relation between regions

To address this second use case, a new entity to group is proposed to group region items and to indicate that the ‘object’ covered by the region defined by the first entity of the group is logically including the areas covered by the regions described by the other entities of the group.

### *Responses to open questions*

First, we recall the semantics implied by region item and masks. According to HEIF specification, the region annotation associated with a region item applies to each region described in the region item individually. If the region is described by a mask (either embedded in the region item or in a separate mask item), the mask indicates which pixels are part of the region, meaning that a mask describes one single region, possibly comprising disjoint areas.

#### Open Question 1

At MPEG#141, the following question has been raised:

* “*Can the Figure 1 in the proposal be achieved for example using mask items where all the regions belonging to a group is part of the mask item?*”

Regarding this question, indeed a mask could represent a region composed of disjoint areas and thus it could be used for representing a region composed of disjoint areas where each area corresponds to a member of the same family. Annotations associated with the region item using that mask then apply to the region described by the mask as a whole and actually would document family information, but additional region items representing each person are still needed to annotate each person individually.

Such approach has following drawbacks compared to the proposed derived region item:

* A mask approach does not allow creating an explicit and direct link between the region item (mask) representing the family and each region item representing each person of the family individually. This relationship may be inferred, e.g., by comparing the pixel areas covered by the masks and each individual region items, but this is more complex for the player than providing an explicit signalling of the relationship.
* A mask approach implies to encode a mask to represent the union of regions representing the family. This is more costly in terms of required data and more complex than using item references to document the union of region items composing the family.

A second alternative was also suggested during MPEG#141 consisting of defining a region item representing the family and comprising a list of regions, one region for each member of the family. As above alternative, this would allow associating annotations (e.g., surname) dedicated to the family with each region individually. But it would still require region items dedicated to each person to associate them with annotations dedicated to the person (e.g., first name). Again, such approach does not allow creating an explicit link between the region item describing the family as a whole and each region item describing each person of the family. This second alternative has same drawbacks as above alternative (including duplicate geometry information needed to describe each region individually and regions in the union).

A third alternative could be to associate both annotations dedicated to the family and annotations dedicated to a person with each region item representing a person.

But this third approach does not allow identifying easily all members of a family. Indeed, this would require to infer the union representing the family by parsing each entry of the ItemPropertyAssociationBox to check whether the corresponding region item is associated with a same annotation identifying the family.

Therefore, we think that the approach based on derived region item represents the best approach to represent a union of regions and to annotate this union as a whole.

#### Open Question 2

At MPEG#141, the following question has been raised:

* “*Is there a restriction for any of the proposals that the separate regions must all be derived from a single image item?*”

It can be noted that region items associated with images representing different visual contents do not share a common referential. Only region items associated with images representing the same visual content have a common referential.

We think that the separate regions referred by the derived region item or the entity group must all be derived from region items sharing a common reference space, i.e., at least from images having a same visual content.

## Proposal

For convenience, we provide below the proposed specification text from TuC with some editorial improvements.

### Proposal 1: Union of regions

Add the following section in section 11.3.3.2 *Derived region item types*

## 11.3.3.2.2 Union derivation

An item with an item\_type value of 'cbrg' defines a derived region item that corresponds to the union of all the regions represented by one or more input region items.

The input region items are specified in a SingleItemTypeReferenceBox of type 'drgn' for this derived region item within the ItemReferenceBox. In the SingleItemTypeReferenceBox of type 'drgn', the value of from\_item\_ID identifies the derived region item of type 'cbrg' and the values of to\_item\_ID identify the input region items.

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

The region resulting from this derived region item is the union of all the regions of each input region item after being applied to the referenced image item as specified in 11.3.2 and 11.2.1.

### Proposal 2: Relations between region items

Add the following section in section 11.3 *Regions and region annotations for an image item*

## 11.3.4 Region Entity Group

## 11.3.4.1 ‘corg’ Entity Group

A compound region entity group ('corg') associates one main region item with one or more region items. It indicates an inclusion relationship between a main object covered by regions of a main entity and other objects covered by regions described by one or more other entities, the main object logically including the other objects.

NOTE For example, a compound region entity group can be used to associate a main region corresponding to a body with regions corresponding to body parts (e.g., the head, legs or arms of the body) to indicate that the body is logically including the body parts.

The entities in a compound region entity group shall be region items. The number of entities in a compound region entity group shall be at least 2. The first entity\_id value shall indicate the main region item. It indicates the region covering the main object that is logically including the objects covered by the regions described by the second and following entity\_ids.

This inclusion relationship does not convey information at the geometry level. A main region signalled as including others regions by a compound entity group may or may not geometrically include the other regions.

# 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.”]]

# On MPEG/JPEG file embedding (MPEG#141, [Issue#87](https://mpeg.expert/software/MPEG/Systems/FileFormat/HEIF/-/issues/87))

## Discussion

During MPEG 140 (cf. [ISOBMFF/Issue#146](https://mpeg.expert/software/MPEG/Systems/FileFormat/isobmff/-/issues/146)), the potential improvement of ISOBMFF 8th edition was extended with a definition of the UUID (see text in section 6.2 below) to enable embedding an ISO base media file within another file. One of these use-cases would be to embed ISOBMFF in JPEG based on JUMBF ISO/IEC 19566-5, which would also allow HEIF files to be embedded into a JPEG file.

At MPEG#141, it was decided to remove the proposed text from ISOBMFF 8th edition for further study in HEIF. It was pointed out that embedding HEIF into JPEG may lead to sub-optimal encapsulation and compatibility issues. Uses cases were also questioned.

## Initial text proposal

*[Ed.(FM): The text below was initially included into potential improvement of ISOBMFF 8th edition clause 6.8 at MPEG#140 and then removed at MPEG#141 for further study]*

**6.8 UUID value for embedded ISO base media files**

When embedding an ISO base media file into a file compliant to another file format that needs a UUID to identify the format of the embedded file, the UUID to identify the ISO base media file shall be equal to 0x49534F30-0011-0010-8000-00AA00389B71.

NOTE This UUID enables embedding an ISO base media file within a file conforming to the JPEG Universal Metadata Box Format (JUMBF, ISO/IEC 19566-5). The JUMBF Content Type in the JUMBF Description box is set equal to the UUID specified above in this subclause. The JUMBF superbox contains a single content box that contains the ISO base media file.

# Disparity adjustment property for frame-packed stereo pair (MPEG #144, issue [#111](https://mpeg.expert/software/MPEG/Systems/FileFormat/HEIF/-/issues/111))

During MPEG #144, a new item property that describes disparity adjustment for a stereo pair entity group was proposed. This property was accepted into the (new) working draft for 3rd edition amendment 2. During the discussion, it was mentioned that this property would also be useful for a frame-packed stereo pair as described by OMAF and that the concept of frame-packed stereo pair items should potentially be moved from OMAF to HEIF.

One objection was raised on this that the current StereoVideoBox in OMAF is overly complicated and a bit wasteful and that rather than simply migrate it from OMAF it might be better to define a new dedicated box.

# SlimHEIF design with Compressed MetaBox (MPEG #146, issue [#151](https://git.mpeg.expert/MPEG/Systems/FileFormat/HEIF/-/issues/151)))

## Abstract

During MPEG #146 the compression of MetaBox with the deflate algorithm for the slimHEIF design was proposed. More than 250 HEIF files for different HEIF usecases were compressed, resulting into an average compressed MetaBox size of approximately 213 bytes.

## Proposal

ISO/IEC 14496-12 supports compression of top-level boxes in subclause 8.19. The processing model when using compression of a top-level box is also well established.

On of the requirement of SlimHEIF design has been to reduce the overhead of HEIF file headers.

In this proposal a solution of compressing the file-level MetaBox with the deflate algorithm was explored. This proposal presents the result of compressing the file-level Metabox against the original file-level MetaBox.

For simulation the conformance files with file-level MetaBox [1][2] was used. The conformance files include different HEIF use cases and is not limited to small images. The file-level MetaBox was compressed using Python zlib library using default settings. The results indicate a saving of 30% on average over 250+ files, whereas in some cases the savings peaked at 65% and above. The average compressed MetaBox size was approximately 213 bytes.

The compression of MetaBox as one of the solutions for the design of SlimHEIF may be considered as it has the following advantages.

* already has a well established processing model
* significant bitrate savings compared to MetaBox
* likely minimal changes to specification text

## References

1. HEIF conformance files Nokia’s GitHub repository <https://github.com/nokiatech/heif_conformance/tree/master/conformance_files>
2. AVIF conformance files

<https://github.com/AOMediaCodec/av1-avif/tree/master/testFiles>

# Increasing Grid Rows and Columns in HEIF (MPEG #147, issue [#160](https://git.mpeg.expert/MPEG/Systems/FileFormat/HEIF/-/issues/160)))

## Abstract

Current efforts are working to utilize HEIF as a format for geospatial imaging applications, where a common configuration hosts imagery in the cloud and remote users access individual tiles in an image pyramid to support panning and zooming across a large geographical area. These applications benefit from accessing image tiles remotely using byte range addressing.

## Use case details

Figure 8.1 shows a very basic diagram of the use case where four tiles are requested from an end user and transferred using four byte range requests, one for each tile.

A diagram of a cloud

Description automatically generated

Figure 8.1: Implementation of remote tile access in HEIF (full image pyramid not shown).

Commercial applications currently exceed 300,000 by 300,000 pixels and the sizes are growing. These images are typically accessed over a network using "cloud optimization" techniques using tiling, image pyramids, and byte range addressing. Tiles are usually set based on end device capabilities. 512x512 or 1024x1024 are common to match handheld and desktop devices. As a user navigates the large image space, individual tiles are pulled depending on pan and zoom commands from the user (typically through a browser interface). HTTP byte range requests are typically used to achieve efficiency and to enable functionality using simple built in browser and web server capabilities. With the current grid parameter settings in HEIF using 8-bit parameters for rows\_minus\_one and columns\_minus\_one, along with a limit of 65535 references via 16-bit reference\_count in the ItemReferenceBox, the limit for the number of tiles in a grid is currently 255x256. This limits the size of grids for this use case to less than current application needs.

Today, cloud optimized geoTIFF is used for this very common use case. HEIF provides significant feature/capability benefits over geoTIFF, so there is significant interest in this facilitating this capability.

## Requirements

Requirements for increasing the size of grid rows and columns within HEIF include:

* Investigate the feasibility of supporting grid images with large image sizes using 32-bit parameters for both grid rows and grid columns (with modest tile sizes, such as 512x512).
* Explore mechanisms to keep the information in the MetaBox needed to support the allocation of a very large number of items acting as the tiles for the grid image to a reasonable size.
* Explore all aspects of updates (number of item references, etc.) required to enable support in HEIF for very large tiled geospatial images.
* Compatibility with and efficient operation using image pyramids.

## Alternative Approaches

To meet basic application requirements when using grids, this action explores the option for expanding the gridding size in HEIF to meet the requirements of this use case and to improve the utility of HEIF in geospatial imaging applications. Alternative options for meeting the requirements also need to be explored. Candidate approaches for performing arrangement of tiles in images and pyramids include:

1. Expanding grid sizes by increasing the size of rows and columns in grids
2. Expanding grid sizes by using grid of grids
3. Using native tiling within a codec and not using grids

During MPEG #147, a few approaches for increasing the size of rows and columns in grids were discussed in the MPEG GitHub (under issue [#105](https://github.com/MPEGGroup/FileFormat/issues/105)) . Few approaches proposed on the GitHub page are repeated here:

### Approach #1:

Images with item\_type='grid' are currently limited to 256x256 tiles because the rows\_minus\_one and columns\_minus\_one are stored as 8 bit integers. Assuming a maximum sensible tile size of 1024x1024 pixels, this means that the largest image resolution is 262144x262144 pixels. This may be too limiting for very large images as they arise, for example, in geospatial imaging.

This approach suggests defining a version=1 in the ImageGrid (ISO 23008-17:2017, Section 6.6.2.3.2) with an option for using larger integers to store rows\_minus\_one and columns\_minus\_one. A possible definition is:

class ImageGrid {

unsigned int(8) version = 1;

unsigned int(8) flags;

FieldLength = ((flags & 1) + 1) \* 16;

TilesFieldLength = ((flags & 2) ? 32 : 8); <<<

unsigned int(TilesFieldLength) rows\_minus\_one; <<<

unsigned int(TilesFieldLength) columns\_minus\_one; <<<

unsigned int(FieldLength) output\_width;

unsigned int(FieldLength) output\_height;

}

Note the new TilesFieldLength that switches the integer size between the old 8 bit and a new 32 bit size. It might also be considered to provide the possibility for a 64bit FieldLength depending on a flag.

### Approach #2:

In this approach, version 0 is used to support smaller grids (existing size) and version 1 when the grid needs to be larger.

aligned(8) class ImageGrid {

unsigned int(8) version;

unsigned int(8) flags;

if (version == 0) {

unsigned int FieldLength = ((flags & 0x01) + 1) \* 16; // this is a temporary, non-parsable variable

unsigned int(8) rows\_minus\_one;

unsigned int(8) columns\_minus\_one;

unsigned int(FieldLength) output\_width;

unsigned int(FieldLength) output\_height;

} else if (version == 1) {

unsigned int FieldLength = ((flags & 0x03) + 1) \* 16; // this is a temporary, non-parsable variable

unsigned int(32) rows\_minus\_one;

unsigned int(32) columns\_minus\_one;

unsigned int(FieldLength) output\_width;

unsigned int(FieldLength) output\_height;

}

}

Where FieldLength is 16 or 32 for the version == 0 case, and 16, 32 or 64 for the version == 1 case. flags equal to 0x02 would not be valid.

Approaches #1 and #2 both support the expansion of rows and columns for grid items but need to address issues related to the expansion of the MetaBox size with the number of allocations required and also the limitation of the number of available item references. Some discussion on this topic can be found on the MPEG GitHub under issue [#106](https://github.com/MPEGGroup/FileFormat/issues/106).

### Approach #3:

This approach suggests using a grid of grids to meet the larger image format requirement. This supports implementation using current capabilities but suffers from more complex decoding logic.

### Approach #4:

This approach suggests using tiling features built into a specific codec. With certain codecs, such as 23001-17, tiling is readily available to utilize in this application and the overhead of reading and parsing the MetaBox is kept to a minimum. The details of utilizing the tiling features of different codecs (HEVC, JPEG 2000, etc.), with their pros and cons needs to be studied. Additional discussion on this topic can be found under this [GitHub issue](https://github.com/strukturag/libheif/issues/1197).

# Generic compression of items (MPEG #147, issue [#157](https://git.mpeg.expert/MPEG/Systems/FileFormat/HEIF/-/issues/157)))

## Abstract

During MPEG #147 the compression of EXIF metadata item was proposed by introducing a new item type for compressed EXIF. However, defining a new item type for a different encoding may not always be a right design choice. Alternative options for meeting the requirements also need to be explored. During MPEG #147 different alternate approaches were discussed in the MPEG GitHub for a generic compression of items(under issue [#157](https://git.mpeg.expert/MPEG/Systems/FileFormat/HEIF/-/issues/157)) .

1. Define a URI that defines the content encoding and use that with a uri item
2. Add a new infe version
3. Extending the mechanism currently under definition for generic compression of uncompressed image items to apply to any uncompressed items (ISO/IEC 23001-17 CDAM2 - [w23515](https://dms.mpeg.expert/doc_end_user/documents/145_OnLine/wg11/MDS23515_WG03_N01148-v3.zip" \t "_blank))