ISO/IEC JTC 1/SC 29/WG 03 N1063

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

**Document type:** Output Document

**Title:** Exploration on low-overhead HEIF-compatible image file format

**Status:** Approved

**Date of document:** 2023-11-03

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

**No. of pages:** 27 (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 N1063**

**October 2023 – Hannover, DE**

|  |  |
| --- | --- |
| **Title** | **Exploration on low-overhead HEIF-compatible image file format** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **23216** |

# Introduction

The High Efficiency Image Format (HEIF) [1] is based on ISOBMFF [2] and provides generic structures for the storage of image items and sequences. These can be compressed using any codec, and HEIF defines metadata necessary for a wide range of applications, making it a perfect foundation for many platforms.

However, as with all technologies, there is always room for improvement. While HEIF's initial design was impressively versatile, it didn't anticipate every potential use-case. A clear inefficiency emerges when considering its header size, which remains large especially for smaller images. This results in unnecessary overhead for such image files. To put this into perspective, GitHub [Issue#59](https://github.com/MPEGGroup/FileFormat/issues/59) highlighted the issue with HEIF headers, which consistently remain at a minimum of 300 bytes. For smaller images, this becomes a significant portion of the file. In this exploration, our goal is to gather and thoroughly evaluate potential solutions to this challenge. Additionally, we'll be identifying relevant use-cases and requirements. Through a systematic approach, we aim to pinpoint the most appropriate solution that enhances the HEIF framework, optimizing it further for compact image formats.

# Use-cases

In this section we collect a potential list of use-cases. Such use cases include:

* **Web Applications with Numerous Icons and Thumbnails**: Modern web applications use a wide variety of icons, thumbnails, and small images. When these images are stored in HEIF, the accumulated overhead from each image's header can result in significantly wasted storage and bandwidth.
* **Mobile Applications**: Many mobile applications frequently utilize small icons and thumbnail images for a better user interface and experience. These applications would greatly benefit from a compact HEIF header, allowing for faster loading times and reduced bandwidth consumption when fetching these images. Additionally, some mobile applications operate in environments with limited bandwidth, making the efficiency provided by a compact HEIF format even more valuable.
* **Embedded Systems and IoT**: Devices with limited storage capabilities, such as certain embedded systems and IoT devices, can benefit from reduced HEIF headers, allowing them to store more images in the same space.
* **Instant Messaging and Social Media Apps**: Users share billions of emojis, stickers, and GIFs daily. If these are stored or shared in HEIF format, even a minor reduction in header size can lead to huge savings in bandwidth and storage on a global scale.
* **Content Delivery Networks and Content Management Systems**: Servers and web deployment services manipulate vast quantities of images, often without decoding the pixels but needing metadata such as image dimensions. A trivially parseable header would save resources at scale and may improve security.
* **Extended Reality (XR) Applications**: XR applications often use small images or icons over real-world views. The efficiency of these small images can be useful for maintaining XR application performance

# Requirements

Below is a list of tentative requirements currently being discussed by the group:

* **Header Size Reduction**: The primary requirement is to substantially reduce the HEIF header size for small images, aiming for a size considerably less than the current 300 bytes.
* **Compatibility**: Even with a reduced header, files should still be identifiable as HEIF.
* **Simplicity**: The header should only cater to the most common use-cases for small images. This includes supporting a single coded item with 1, 3, and potentially 4 channels. It should also support an optional second coded item for alpha and NCLX colour.
* **Exclusion of Non-Essential Features**: The base format of reduced header mode should omit non-essential features like tiling/overlay, image collections/sequences, auxiliary images (excluding alpha), and groups. It can be constructed in a manner that allows for future extensibility to incorporate additional features that are reasonable for such a reduced header representation.
* **Optional Exclusion of Additional Features**: If these requirements do not significantly impact the header size, they may be considered. However, for utmost efficiency, thumbnails and other non-essential metadata should probably be excluded.
* **Structured Layout for Streamed Decoding:** For optimized streamed or incremental decoding, the format might enforce a specific sequence, such as placing alpha first.
* **Round-tripping**: It should be possible to translate the compact HEIF representation into non-compact representation, and back.
* **Mixed Reduced Header HEIF and ISOBMFF**: It would be desirable to allow the presence of reduced header HEIF and ISOBMFF-compliant media in the same file. For example, a file could contain a reduced-header HEIF image and an associated audio track to be played with the image.

The group is currently focusing on the aspect of compatibility to answer the question if the HEIF file with a minimized top level box that replaces the Metabox is still considered as a HEIF compatible file. It was noted that a very similar approach was already implemented in WG3 during the development of the 7th edition of ISOBMFF (ISO/IEC 14496-12 6th edition DAM 4). This involved the concept of compressing top-level boxes, such as substituting the 'moov' box with a compressed '!mov' box which mirrors the proposal from section 4.2. It is worth noting that when ISOBMFF was extended to define compressed boxes, the process remained within the domain of WG3 and was deemed an improvement on existing technology developed by WG3 and did not require the involvement of the requirements working group, which is generally not engaged in incremental optimizations of a standard.

Furthermore, the shall requirements of the HEIF and ISOBMFF specifications were studied with the following findings:

1) The existing language in the HEIF specification mandates the inclusion of a Metabox at the file level. It has been recognized that permitting the inclusion of a minified version of this box would necessitate only minimal modifications to the specification.

It has been discussed that the minimized top-level box may be interpreted or treated as analogous to the Metabox, implying its virtual presence within a file. Additionally, it has been acknowledged that any extension to the current Metabox would necessitate a revision of the ISOBMFF. Given the existing discrepancy in the Metabox structure between ISOBMFF and QuickTime File Format, pursuing this method is deemed to have a reduced likelihood of success.

2) It was identified that if a minimized top-level box is introduced, the mime type image/heif can not be used. However most derived specifications use their own mime types. Additionally, the 'mini' box could only be used in future, not-yet-registered mime types (HEIF+AV2 specification for example) to avoid breaking any existing mime type and to avoid registering new mime types for existing container/codec pairs.

Also, a requirements document on image items was studied [6] and the following changes were agreed to be included in the output document from WG2:

**In section 1.1.1, change the sentence:**

Size taken by the picture matters as it relates to upload duration as well as the amount of transferred bytes.

**To:**

Size taken by the picture matters as it relates to upload and download duration as well as the amount of transferred bytes.

**In section 1.2.5, change the sentence:**

The goal is to view the image in as high of a resolution/quality as possible, e.g. 16-bit lossless (lossless is mandatory in several categories of medical usage).

**To**

The goal is to view the image in as high of a resolution/quality as possible, e.g. 16-bit lossless (lossless is mandatory in several categories of medical usage, Monochrome/4:0:0 is a common color format).

**After requirement 6 in section 2, add the following new requirement and renumber the requirements under Related to Coding: as needed:**

7. HEVC still pictures should be competitive with regards to compression ratio compared to existing image file formats.

# Initial proposals

At MPEG#143, the File Format group received two distinct proposals aiming to resolve this problem. These are detailed in the following subsections.

## MetaBox extension and a new normative Annex

Nokia, in [m64322](https://dms.mpeg.expert/doc_end_user/current_document.php?id=88666&id_meeting=195), introduced a proposal centered on the extension of the MetaBox which allows to create a file without an ItemLocationBox, and a definition of a new normative Annex, which describes the reduced header mode including the file structure and the reader/player operation for a single image item and an optional auxiliary image item.

Initial discussion on this proposal is gathered in the [GitLab issue #104](https://mpeg.expert/software/MPEG/Systems/FileFormat/HEIF/-/issues/104).

### Proposal

#### Changes to MetaBox

Define MetaBox with a new version=1. This version allows to create a file without an ItemLocationBox and requires readers to handle files without ItemLocationBox.

##### Definition

Box Type: 'meta'  
Container: File, Segment, MovieBox, TrackBox, MovieFragmentBox or TrackFragmentBox  
Mandatory: No  
Quantity: Zero or one (in File, MovieBox, and TrackBox),  
 Zero or one (in Segment, MovieFragmentBox or TrackFragmentBox)

A common base structure is used to contain general untimed metadata. This structure is called the MetaBox as it was originally designed to carry metadata, i.e. data that is annotating other data. However, it is now used for a variety of purposes including the carriage of data that is not annotating other data, especially when present at ‘file level’. The handling of metadata in movie fragments is described in 8.8.17.

When a HandlerBox is present, it applies to all items without a HandlerProperty and may provide additional requirements on items with a HandlerProperty with different handler\_type than the one in the HandlerBox.

When the MetaBox contains a PrimaryItemBox and a HandlerBox, and the item indicated by the PrimaryItemBox has a HandlerProperty, the HandlerBox and the HandlerProperty of the primary item shall identify the same handler type.

When the MetaBox does not contain a PrimaryItemBox, then MetaBox is required to contain a HandlerBox indicating the structure or format of the MetaBox contents.

When the item indicated by PrimaryItemBox does not have a HandlerProperty, but has an ItemInfoEntry with an item\_type, the handler type in HandlerBox may be the same as the item\_type.

The other boxes defined here may be defined as optional or mandatory for a given format. If they are used, then they shall take the form specified here. These optional boxes include a DataInformationBox, which documents other files in which metadata values (e.g. pictures) are placed, and an ItemLocationBox, which documents where in those files each item is located (e.g. in the common case of multiple pictures stored in the same file).

At most one MetaBox may occur at each of the file level, segment, movie level, or track level.

If an ItemProtectionBox occurs, then some or all of the metadata, including possibly the primary resource, may have been protected and be un-readable unless the protection system is taken into account.

NOTE The MetaBox is unusual in that it is a container box yet extends FullBox, not Box.

Metadata items are identified by item\_ID. Within a given MetaBox, a given item\_ID shall uniquely refer to a single item. When an item is updated in movie fragments, the item\_ID refers to the latest received version.

Derived specifications may further restrict the criteria for uniqueness: unique among the item\_IDs in both file and movie-level boxes, or unique within that set extended with the track\_ID of the tracks in a movie box. The item\_ID value of 0 should not be used, and shall not be used when the set is extended to include track\_IDs.

There are three scopes for item\_IDs: file and segments; MovieBox and MovieFragmentBox; and TrackBox and TrackFragmentBox. In other words, there shall be only one item with a given item\_ID within a given scope (e.g. in the TrackBox and all TrackFragmentBox with the same track\_ID).

version shall not be equal to 1 in a movie-level or track-level MetaBox.

When version is equal to 1 and the ItemLocationBox is absent, the file shall obey the following constraints:

The file shall contain one and only one MediaDataBox or IdentifiedMediaDataBox per each item.

* The file shall not contain MediaDataBox(es) or IdentifiedMediaDataBox(es) that contain data other than item data.
* The order of MediaDataBox(es) and IdentifiedMediaDataBox(es) in the file shall be in ascending order of item IDs.
* The item data for the primary item shall be present in the first MediaDataBox (when present) or IdentifiedMediaDataBox (when present), whichever is earlier in the file.
* Each item shall have only one extent.
* There shall be no other data than the item data in the MediaDataBox or IdentifiedMediaDataBox.

When version is equal to 1 and the ItemLocationBox is absent, a file reader shall resolve the item data as follows:

* The list of item ID values in ascending order is obtained from the ItemInfoBox.
* The item data for the first item in the list of item ID values is located in the first MediaDataBox (when present) or IdentifiedMediaDataBox (when present), whichever is earlier in the file. Item data for each subsequent item in the list of item ID values is resolved to be the box payload of the next MediaDataBox or IdentifiedMediaDataBox in file order.

NOTE MetaBox with version equal to 1 can be used in an item file, such as a HEIF file with a single image item, to avoid the overhead for an ItemLocationBox.

##### Syntax

aligned(8) class MetaBox (handler\_type)  
 extends FullBox('meta', version, 0)   
{  
 HandlerBox(handler\_type) theHandler;  
 PrimaryItemBox primary\_resource; // optional  
 DataInformationBox file\_locations; // optional  
 ItemLocationBox item\_locations; // optional  
 ItemProtectionBox protections; // optional  
 ItemInfoBox item\_infos; // optional  
 IPMPControlBox IPMP\_control; // optional  
 ItemReferenceBox item\_refs; // optional  
 ItemDataBox item\_data; // optional  
 Box other\_boxes[]; // optional  
}

#### Guidelines for reduced header mode (Annex O)

(normative)  
**Guidelines for reduced header mode**

**Overview**

This annex gives guidelines to enable the compact item header or reduced header mode in file structures and the reader/player operation for reduced header mode. The reduced header mode enables storage of a single image item and optionally an auxiliary image item for alpha planes with a compact representation of the image file format.

**File structure**

The following file creation guidelines enable the reduced header mode.

The brands with which a file is compatible are recorded in the file in the usual way using the FileTypeBox. The file-level MetaBox should precede the MediaDataBox(es) or IdentifiedMediaDataBox(es), whichever is earlier in file. The file-level MetaBox shall be with version=1.

The MetaBox documents the information related to a single image item and optionally an auxiliary image item for alpha planes.

The MetaBox contains the ItemProtectionBox if either the single image item or the auxiliary image item or both are protected.

The MetaBox contains the ItemInfoBox, which provides information about the single image item and optionally an auxiliary image item for alpha planes. The item\_IDs in ItemInfoBox uniquely refer to image items in MetaBox.

When both the single image item and the auxiliary image item for alpha planes are present in MetaBox the item\_IDs in ItemInfoBox are set such that the single image item has the lowest item\_ID value among the image items.

When only a single image item or both the single image item and the auxiliary image item for alpha planes are present in MetaBox they have only one extent.

When only a single image item is present, the file contains one and only one MediaDataBox or IdentifiedMediaDataBox, the item data for the item is present in the MediaDataBox or IdentifiedMediaDataBox, and there is no other data than the item data in the MediaDataBox or IdentifiedMediaDataBox.

When both the single image item and the auxiliary image item for alpha planes are present, the file contains two MediaDataBoxes or two IdentifiedMediaDataBoxes. The MediaDataBox or IdentifiedMediaDataBox of the primary item should precede the MediaDataBox or IdentifiedMediaDataBox of the auxiliary item in the file.

The MetaBox contains the ItemPropertiesBox to associate items with item properties.

**Reader/Player operation**

This clause provides guidelines for readers/players that use reduced header mode.

If a file contains the MetaBox with version=1, then a reader/player concludes that the file is in reduced header mode

When a file is in reduced header mode and does not contain the ItemLocationBox the reader/player concludes that the items have only one extent. The reader/player obtains the list of item\_ID values in ascending order from the ItemInfoBox. The reader/player obtains the item data for the first item from the list of item\_ID values from the first MediaDataBox (when present) or IdentifiedMediaDataBox (when present), whichever is earlier in the file. The reader/player obtains the item data for each subsequent item in the list of item\_ID values from the resolved box payload of the next MediaDataBox or IdentifiedMediaDataBox in file order.

## Minimized image item

In [m64572](https://dms.mpeg.expert/doc_end_user/current_document.php?id=88916), Apple Inc. and Google LLC first proposed a different approach. They introduced a new minimized image item (initially called condensed image item) designed to represent the essential information for small images, with the goal to maintain the format's performance and capability. That essential information can be used to expand the minimized image item into a regular HEIF file.

Initial discussion on this proposal is gathered in the [GitLab issue #105](https://mpeg.expert/software/MPEG/Systems/FileFormat/HEIF/-/issues/105).

A prototype of this proposal can be found in this [pull request](https://github.com/AOMediaCodec/libavif/pull/1432).

During MPEG meeting #144 further refinements and clarifications were made in [m64748](https://dms.mpeg.expert/doc_end_user/current_document.php?id=89369) and discussed in [GitLab issue #106](https://mpeg.expert/software/MPEG/Systems/FileFormat/HEIF/-/issues/106) as well as during the MPEG meeting. The proposal below is updated accordingly.

### Overview

The following are example payload sizes for a medium quality image at various resolutions:

* 640x480: 10.5 kB
* 320x240: 4.4 kB
* 160x120: 1.7 kB
* 80x60: 0.6 kB
* 40x30: 0.3 kB

The current HEIF file structure adds around 300 bytes of headers to the payload. For example, for images with a 40x30 resolution, this could lead to a ~2x file-size increase [4].

This contribution proposes a new Minimized Image Item Box ('mini'), which is intended to minimize data overhead in the structure of a HEIF file. The goal is to allow minimal overhead for the following very common file types:

* 1/3-channel opaque images
* 1/3-channel translucent images with alpha (supporting codecs with native alpha channel support that can handle interleaved alpha and codecs without native alpha channel support requiring a separate auxiliary alpha image)
* Images with Exif and XMP metadata
* Images with NCLX or ICC profiles

Transformative item properties like 'imir', 'irot', and 'clap' *can* be used via the 'has\_extended\_meta' field. But since the main goal of this contribution is to achieve small file sizes, rotation and cropping should be baked in rather than done at decode time. Therefore, they don't get explicit fields in the syntax structure defined.

Below is an example file structure, which shows how the MinimizedImageBox is used in a file:

{  
 ('ftyp' "File Type Box", size = 20) {  
 Major brand: 'abcd'  
 Minor version: 0  
 Compatible brands: 'abcd'  
 }  
 ('mini' "Minimized Image Item Box", size = nnn) {  
 ...  
 }  
 (Optional 'moov'/'mdat')  
}

NOTE It is currently being studied in ISOBMFF TuC if further optimizations on the 'ftyp' box are feasible or not. The current study investigates if the first four bytes of the 'mini' box body can be moved to the minor\_version field of the 'ftyp' box and the size of the 'mini' box is reduced by four bytes.

The 'mini' box in this structure serves as the primary container for the image-specific data, housing everything from colour characteristics to codec configurations, and from alpha channel presence to the image data itself.

The file begins with the 'ftyp' box that carries a brand identifier. To further compress and prevent redundant signalling, the major brand can implicitly signal the codec type and a codec configuration type. This concept can be employed by derived specifications to define their own presets. However, if no codec specific brand exists, the 'mini' brand may be used, in which case 'has\_explicit\_codec\_types' shall be set to true. This allows for the box to be codec agnostic but also allows optimized codec specific brands to save 8 bytes.

The optional 'moov'/'mdat' boxes can be used when a flag has\_extended\_meta is set to true and allows for adding additional image items and/or grouping image items to tracks.

### Experimental results based on initial proposal from [m64572](https://dms.mpeg.expert/doc_end_user/current_document.php?id=88916)

Below is an example image with a 40x30 P3D65 [3] payload of 300 bytes that has a codec specific ‘mini’ brand, the overhead on top of the payload and codec config box is:

ftyp: 20 bytes  
 - 8 bytes box header  
 - 4 bytes major brand (codec specific 'mini' brand)  
 - 4 bytes minor version  
 - 4 bytes compatible brands (codec specific 'mini' brand repeated)  
coni: 17 bytes  
 - 64 bits (8 bytes) for box header  
 - 2 bits version  
 - 8 bits width  
 - 8 bits height  
 - 1 bits is\_float  
 - 4 bits bitdepth  
 - 1 bits is\_monochrome  
 - 1 bits is\_full\_range  
 - 2 bits colour\_type  
 - 15 bits NCLX  
 - 1 bits has\_explicit\_codec\_types  
 - 8 bits mainItemCodecConfigSize  
 - 16 bits mainItemDataSize  
 - 1 bits has\_alpha  
 - 1 bits has\_extended\_meta  
 - 1 bits has\_exif  
 - 1 bits has\_XMP  
 - 1 bits trailing to get byte alignment

The table below provides further example payload sizes for various resolutions:

|  |  |  |  |
| --- | --- | --- | --- |
| **Compressed size (8-bit 4:2:0)** | **Lossless** | **CQ 16** | **CQ 32** |
| **640x480** | 152.6K | 24.7K | 10.5K |
| **320x240** | 41.9K | 8.6K | 4.4K |
| **160x120** | 12.6K | 3.1K | 1.7K |
| **80x60** | 3.9K | 1.2K | 0.6K |
| **40x30** | 1.2K | 0.4K | 0.3K |
|  |  |  |  |
| **% increase 300 bytes headers** |  |  |  |
| **640x480** | 0.2% | 1.2% | 2.8% |
| **320x240** | 0.7% | 3.5% | 6.8% |
| **160x120** | 2.4% | 9.6% | 17.8% |
| **80x60** | 7.6% | 25.4% | 47.1% |
| **40x30** | 24.9% | 67.9% | 111.5% |
|  |  |  |  |
| **% increase 40 bytes headers** |  |  |  |
| **640x480** | 0.0% | 0.2% | 0.4% |
| **320x240** | 0.1% | 0.5% | 0.9% |
| **160x120** | 0.3% | 1.3% | 2.4% |
| **80x60** | 1.0% | 3.4% | 6.3% |
| **40x30** | 3.3% | 9.0% | 14.9% |

### Proposal

#### Minimized Image Item Box

##### Definition

|  |  |
| --- | --- |
| Box type: | 'mini' |
| Container: | file |
| Mandatory: | No |
| Quantity: | At most one |

The minimized image item box provides a more compact way to represent carriage of image items in a file. Its main use case is for very small images where the usage of traditional carriage using the MetaBox would result in considerable overhead compared to the image data payload.

When MinimizedImageBox is present, a file-level MetaBox shall not be present in the file. However, some parts of the body of a MetaBox may be embedded in the MinimizedImageBox when the has\_extended\_meta flag is set to one.

The major\_brand of the FileTypeBox may be specified in derived specifications to signal pre-defined values for infe\_type and codec\_config\_type. However, if no such codec specific brand exists, the 'mini' brand may be used, in which case has\_explicit\_codec\_types shall be set to 1.

A file that contains a MinimizedImageBox shall have major\_brand of the FileTypeBox set to 'mini', or to a derived specification’s brand that conforms to the 'mini' brand.

The MinimizedImageBox may be followed by a MovieBox.

The MinimizedImageBox may be followed by a MediaDataBox.

A file with a MinimizedImageBox shall be expanded to a full file-level MetaBox, treated as if the MetaBox was originally present. The rules for expanding MinimizedImageBox to a full MetaBox are described in clause x.x.x.5.

##### Syntax

aligned(8) class MinimizedImageBox extends Box('mini') {  
 bit(2) version;  
 sqlite\_varint width\_minus\_one;  
 sqlite\_varint height\_minus\_one;

// Colour and bit-depth  
 bit(1) is\_float;  
 if (is\_float) {  
 bit(2) float\_precision;  
 }  
 else {  
 bit(4) bit\_depth\_minus\_one;  
 }  
 bit(2) subsampling;  
 if (subsampling >= 2) {  
 bit(1) is\_centered;  
 }  
 bit(1) full\_range; // TBD integer based in CICP, exclude if float  
 bit(2) colour\_type;  
 if (colour\_type == 0) {  
 colour\_primaries = 1;  
 transfer\_characteristics = 13;  
 matrix\_coefficients = 6;  
 }  
 else if (colour\_type == 1) {  
 bit(5) colour\_primaries;  
 bit(5) transfer\_characteristics;  
 if (subsampling > 0) { // if not monochrome  
 bit(5) matrix\_coefficients;  
 }  
 else {  
 matrix\_coefficients = 2;  
 }  
 }  
 else if (colour\_type == 2) {  
 bit(8) colour\_primaries;  
 bit(8) transfer\_characteristics;  
 if (subsampling > 0) { // if not monochrome  
 bit(8) matrix\_coefficients;  
 }  
 else {  
 matrix\_coefficients = 2;  
 }  
 }  
 else {  
 colour\_primaries = 2;  
 transfer\_characteristics = 2;  
 if (subsampling > 0) { // if not monochrome  
 bit(8) matrix\_coefficients;  
 }  
 else {  
 matrix\_coefficients = 2;  
 }  
 sqlite\_varint icc\_data\_size\_minus\_one;  
 }

// Item metadata  
 bit(1) has\_explicit\_codec\_types;  
 if (has\_explicit\_codec\_types) {  
 unsigned int(32) infe\_type;  
 unsigned int(32) codec\_config\_type;  
 }  
 sqlite\_varint main\_item\_codec\_config\_size;  
 sqlite\_varint main\_item\_data\_size\_minus\_one;

// Other items  
 bit(1) has\_alpha;  
 if (has\_alpha) {  
 bit(1) alpha\_is\_premultiplied;  
 sqlite\_varint alpha\_item\_codec\_config\_size;  
 sqlite\_varint alpha\_item\_data\_size;  
 }  
 Boolean has\_separate\_alpha\_item = has\_alpha && alpha\_item\_data\_size > 0;  
 bit(1) has\_extended\_meta;  
 if (has\_extended\_meta) {  
 sqlite\_varint extended\_meta\_size\_minus\_one;  
 }  
 bit(1) has\_exif;  
 if (has\_exif) {  
 sqlite\_varint exif\_data\_size\_minus\_one;  
 }  
 bit(1) has\_xmp;  
 if (has\_xmp) {  
 sqlite\_varint xmp\_data\_size\_minus\_one;  
 }

// Pad bits until byte-aligned  
 trailing\_bits();

// Payload data

// Codec config body data for alpha and main  
 if (has\_alpha && alpha\_item\_codec\_config\_size > 0) {  
 unsigned int(8) alpha\_item\_codec\_config[alpha\_item\_codec\_config\_size];  
 }  
 unsigned int(8) main\_item\_codec\_config[main\_item\_codec\_config\_size];

// Extended 'meta' box  
 if (has\_extended\_meta) {  
 unsigned int(8) extended\_meta[extended\_meta\_size\_minus\_one + 1];  
 }

// ICC profile data  
 if (colour\_type == 3) {  
 unsigned int(8) icc\_data[icc\_data\_size\_minus\_one + 1];  
 }

// Alpha and main elementary stream payloads  
 if (has\_separate\_alpha\_item) {  
 unsigned int(8) alpha\_data[alpha\_item\_data\_size];  
 }

unsigned int(8) main\_data[main\_item\_data\_size\_minus\_one + 1];

// Metadata payloads  
 if (has\_exif) {  
 unsigned int(8) exif\_data[exif\_data\_size\_minus\_one + 1];  
 }

if (has\_xmp) {  
 unsigned int(8) xmp\_data[xmp\_data\_size\_minus\_one + 1];  
 }  
}

##### Semantics

version: version of the MinimizedImageBox. The current version shall be set to 0.

width\_minus\_one: specifies the width minus one of the reconstructed image in pixels, as specified in ImageSpatialExtentsProperty in clause 6.5.3

height\_minus\_one: specifies the height minus one of the reconstructed image in pixels, as specified in ImageSpatialExtentsProperty in clause 6.5.3

is\_float: specifies whether float\_precision or bit\_depth\_minus\_one is signalled. If is\_float is set to 1, it indicates that the float\_precision is signalled, otherwise bit\_depth\_minus\_one is signalled.

float\_precision: specifies the format of floating-point numbers used for the pixel values as defined by IEEE 754-2008. The values 0, 1, and 2 correspond to half-precision float (binary16), single-precision float (binary32), and double-precision float (binary64) formats, respectively. Other values are reserved for a future specification. When is\_float is set to 0, the value is undefined.

bit\_depth\_minus\_one: indicates the number of bits, minus one, per channel for the pixels of the reconstructed main and alpha image items, as specified in PixelInformationProperty in clause 6.5.6.

subsampling: when set to 0, indicates that there is exactly one channel of coded colour samples, as specified by the num\_channels field of the PixelInformationProperty in clause 6.5.6. When set to a non-zero value it indicates that there are exactly three channels of coded colour samples. A value of 1 indicates that there is no subsampling of chroma (i.e. 4:4:4). A value of 2 indicates that chroma is subsampled by a factor 2 horizontally (i.e. 4:2:2). A value of 3 indicates that chroma is subsampled both horizontally and vertically by a factor 2 (i.e. 4:2:0).If has\_alpha is 1 and alpha\_item\_codec\_config\_size is 0, the number of channels will be two and four respectively.  
EDITORS NOTE: Add language to indicate how this translates to the newly proposed `pixi` that contains subsampling.

is\_centered: 0 indicates that the chroma samples are co-located with the luma samples. A value of 1 indicates that the chroma samples are centered between the luma samples.   
EDITORS NOTE: Add language to indicate how this translates to the newly proposed `pixi` that contains subsampling.

full\_range: carries a VideoFullRangeFlag value as defined in ISO/IEC 23091-2

colour\_type: specifies the colour encoding type. When set to 0 it indicates the default values of MIAF (1/13/6). When set to 1 or 2 it implies the on-screen colours as signalled in ColourInformationBox with colour\_type='nclx'. When set to 3 it indicates that an ICC Profile and matrix coefficients are present.

colour\_primaries: carries a ColourPrimaries value as defined in ISO/IEC 23091-2

transfer\_characteristics: carries a TransferCharacteristics value as defined in ISO/IEC 23091-2

matrix\_coefficients: carries a MatrixCoefficients value as defined in ISO/IEC 23091-2

icc\_data\_size\_minus\_one: specifies the size of ICC profile data minus one when the colour\_type field indicates it is present in bytes. Undefined if the value of colour\_type is not equal to 3.

has\_explicit\_codec\_types: when set to 1 indicates that both infe\_type and codec\_config\_type are explicitly signalled, otherwise their types are implied from the major\_brand of the FileTypeBox. Shall be set to 1 if major\_brand does not explicitly specify their default values.

infe\_type: corresponds to the item\_type field of the version 2 of the ItemInfoEntry box. Defined by the major brand if has\_explicit\_codec\_types is set to 0.

codec\_config\_type: corresponds to the codec configuration box type. Defined by the major brand if has\_explicit\_codec\_types is set to 0.

main\_item\_codec\_config\_size: specifies the size of the configuration for the main image item.

main\_item\_data\_size\_minus\_one: specifies the size minus one of the data for the main image item in bytes.

has\_alpha: when set to 0 indicates that the image is opaque, otherwise the image has an alpha layer, whether the codec has native translucency support or an auxiliary image item is used.

alpha\_is\_premultiplied: when set to 1 indicates that main values are pre-multiplied by alpha, otherwise main values are not pre-multiplied.

alpha\_item\_codec\_config\_size: specifies the size of the configuration for the alpha image item in bytes. When set to 0 indicates that the codec does not need any configuration data for alpha or can reuse the one from the main image. The value is set to 0 if has\_alpha is 0.

alpha\_item\_data\_size: specifies the size of the data for the alpha image item in bytes. If has\_alpha is set to 1, the value 0 indicates that the codec has native translucency support and that the alpha samples are coded alongside the colour samples in the main\_data chunk. Zero if has\_alpha is not set to 1.

has\_extended\_meta: when set to 1 indicates the presence of extended metadata within the MinimizedImageBox, otherwise it indicates the absence of it.

extended\_meta\_size\_minus\_one: specifies the size minus one of the extended metadata in bytes. Undefined if has\_extended\_meta is not set to 1.

has\_exif: when set to 1 indicates the presence of an Exif metadata chunk, otherwise it indicates the absence of it.

exif\_data\_size\_minus\_one: specifies the size minus one of the Exif metadata in bytes. Undefined if has\_exif is not set to 1.

has\_xmp: when set to 1 indicates the presence of an XMP metadata chunk, otherwise it indicates the absence of it.

xmp\_data\_size\_minus\_one: specifies the size minus one of the XMP metadata in bytes. Undefined if has\_xmp is not set to 1.

trailing\_bits: padding bits to ensure payloads are 8-bit aligned. Shall be 0.

alpha\_item\_codec\_config: specifies the optional alpha image codec configuration data. When has\_alpha is set to 0 or alpha\_item\_codec\_config\_size is 0, alpha\_item\_codec\_config is not present.

main\_item\_codec\_config: specifies the main image item codec configuration data. When main\_item\_codec\_config\_size is 0, main\_item\_codec\_config is not present.

extended\_meta: specifies the extended metadata that may be optionally included within a file. When the has\_extended\_meta flag is set to 0, the extended\_meta field is not present. Conversely, if the extended\_meta is present within a file, the readers shall support and correctly process the extended\_meta field.

icc\_data: specifies the optional ICC profile data. When colour\_type is not set to 3 icc\_data is not present.

alpha\_data: specifies the optional alpha image data. When has\_alpha is set to 0 or alpha\_item\_data\_size is 0, alpha\_data is not present.

main\_data: specifies the main image data.

exif\_data: specifies the optional Exif metadata. When has\_exif is set to 0 exif\_data is not present.

xmp\_data: specifies the optional XMP metadata. When has\_xmp is set to 0 xmp\_data is not present.

The data type sqlite\_varint will also need to be defined in HEIF. Its original definition can be found [here](https://www.sqlite.org/src4/doc/trunk/www/varint.wiki). However, it was noted that its definition could be further modified to better suit the use-case for SlimHEIF and its initial design could look like this:

aligned(8) class SQLiteSlimVarInt {  
 unsigned int(8) a0;  
 if(a0 > 245) {  
 unsigned int(8) a1;  
 }  
 if(a0 > 253) {  
 unsigned int(8) ax[a0-253]; // 1 or 2 bytes  
 }

// the following is non parsable stuff to show how to compute size  
 int outputValue = 0; // non-parable variable  
 switch(a0) {  
 case 0 ... 245:  
 outputValue = a0;  
 break;  
 case 246 ... 253:  
 outputValue = 246+256\*(a0-246)+a1;  
 break;  
 case 254:  
 outputValue = 2294+256\*a1+ax[0];  
 break;  
 case 255:  
 outputValue = a1<<16 + ax[0]<<8 + ax[1];  
 break;  
 }  
};

The SlimHEIF optimized version of sqlite\_varint can be determined based on the [following data](http://mpeg.expert/software/MPEG/Systems/FileFormat/HEIF/-/issues/106" \l "note_83456) gathered from the web images. For all fields where sqlite\_varint is used we find the following values for the sizes of the fields:

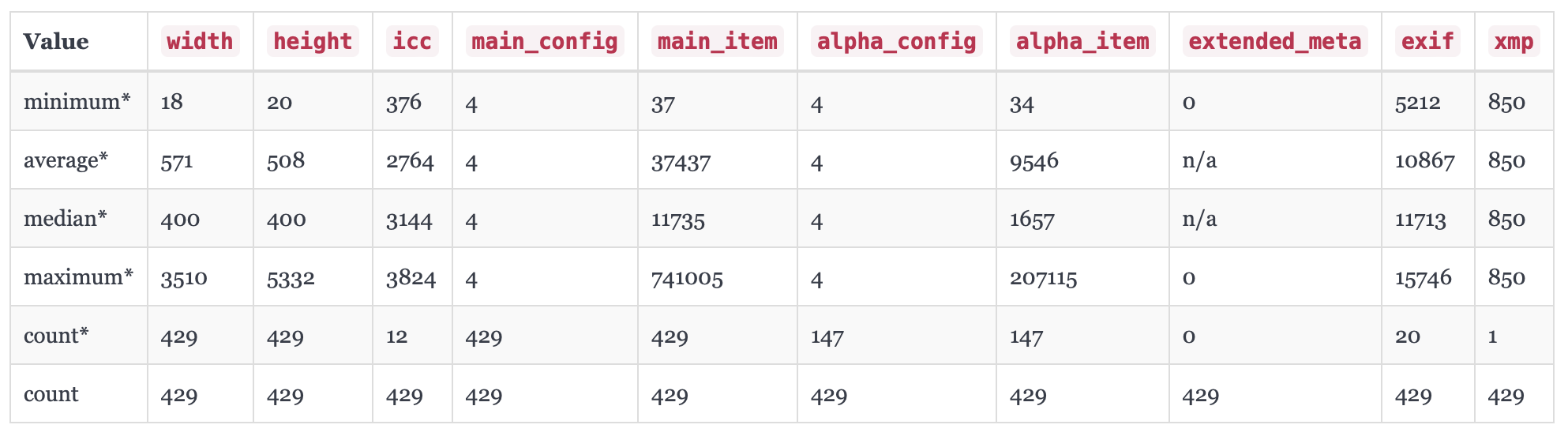


Figure 1: Results on 429 curated PNGs with various natural and artificial contents

A screenshot of a white table

Description automatically generated

Figure 2: Results on 50k PNGs from the Web

A screenshot of a white table

Description automatically generated

Figure 3: Results on 100k JPEGs from the Web

extended\_meta is likely used for EXIF rotation only.

\*among non-zero values

Based on this the optimized allocation of bits in SQLiteSlimVarInt can be determined.

#### x.x.x.4 extended\_meta payload requirements

The extended\_meta payload may contain any of the boxes normally allowed in a file-level MetaBox with the following restrictions:

* It shall have no HandlerBox
* It shall have no DataInformationBox
* It shall have no ItemDataBox
* It shall have no ItemInfoBox entries for items 1 to 16
* It shall not have ItemPropertyAssociationBox box entries for items 1 to 16 that add associations to item property indices 1 to 16.
* It may have an ItemLocationBox. If present, any offsets specified shall be offsets in the actual file, not after applying 1.1.1.5 MetaBox equivalence.
* It shall have no ItemLocationBox entries for items 1 to 16
* It shall have no ItemLocationBox entries for other items using construction method 1.

It shall have no ItemReferenceBox entry that adds an alpha auxiliary item to item 1.

#### x.x.x.5 MetaBox equivalence

A MinimizedImageBox has a one-to-one mapping to a full MetaBox. This means that parsers processing a MinimizedImageBox shall treat it as if it was replaced by this equivalent full MetaBox.

This expanded MetaBox shall have version = 0 and flags = 0. The expanded MetaBox shall have the sub-boxes described by the following sections.

1. **HandlerBox**

The expanded MetaBox shall have a HandlerBox with handler\_type equal to 'pict'.

1. **PrimaryItemBox**

If extended\_meta contains a PrimaryItemBox, this box is used in the expanded MetaBox.

If no such box is present, the expanded MetaBox shall have a PrimaryItemBox with item\_ID set to 1.

1. **ItemInfoBox**

The expanded MetaBox shall have an ItemInfoBox containing the following entries:

* ItemInfoEntry of version 2 with item\_ID set to 1 and item\_type set to infe\_type. All other fields are set to null or 0 as appropriate.
* If has\_separate\_alpha\_item is 1, ItemInfoEntry with item\_ID set to 2 and item\_type set to infe\_type. All other fields are set to null or 0 as appropriate.
* If has\_exif is 1, ItemInfoEntry with item\_ID set to 3 and item\_type set to Exif. All other fields are set to null or 0 as appropriate.
* If has\_xmp is 1, ItemInfoEntry with item\_ID set to 4 and item\_type set to mime and content\_type set to 'application/rdf+xml'. All other fields are set to null or 0 as appropriate.

If extended\_meta contains an ItemInfoBox, its entries shall be appended to the entries described above.

1. **ItemReferenceBox**

The ItemReferenceBox is populated with the following entries:

* If ItemInfoBox has an entry for item\_ID 2: Item type reference with referenceType set to 'auxl', reference\_count set to 1, from\_item\_ID set to 2 and to\_item\_ID set to 1.
* If ItemInfoBox has an entry for item\_ID 2 and alpha\_is\_premultiplied is set to 1: Item type reference with referenceType set to 'prem', reference\_count set to 1, from\_item\_ID set to 1 and to\_item\_ID set to 2.
* If ItemInfoBox has an entry for item\_ID 3: item type reference with referenceType set to 'cdsc', reference\_count set to 1, from\_item\_ID set to 3 and to\_item\_ID set to 1.
* If ItemInfoBox has an entry for item\_ID 4: item type reference with referenceType set to 'cdsc', reference\_count set to 1, from\_item\_ID set to 4 and to\_item\_ID set to 1.

If extended\_meta contains an ItemReferenceBox, its entries shall be appended to the entries described above.

If the resulting ItemReferenceBox contains at least one entry, it shall be added to the expanded MetaBox. An empty ItemReferenceBox shall be ignored.

1. **ItemPropertiesBox**

The expanded MetaBox shall have an ItemPropertiesBox containing an ItemPropertyContainerBox and an ItemPropertyAssociationBox.

The ItemPropertyContainerBox shall have 16 entries as listed below. Any entry for which the condition is not true is replaced with a FreeSpaceBox.

|  |  |  |
| --- | --- | --- |
| Entry | Condition | Contents |
| 1 | main\_item\_codec\_config\_size is not 0 | Item property with the type set to codec\_config\_type and with contents from main\_item\_codec\_config |
| 2 | true | ImageSpatialExtentsProperty with image\_width set to width\_minus\_one + 1 and image\_height set to height\_minus\_one + 1 from the MinimizedImageBox |
| 3 | is\_float is 0 | PixelInformationProperty with num\_channels set to   * 1 if is\_monochrome is 1 * 3 otherwise   and bits\_per\_channels set to bit\_depth\_minus\_one+1.  NOTE: This should be updated once we have an accepted proposal for GitHub issues #69 and #70. |
| 4 | true | ColourInformationBox with colour\_type set to 'nclx' and colour\_primaries, transfer\_characteristics, matrix\_coefficients and full\_range set to the values from the MinimizedImageBox. |
| 5 | colour\_type is 3 | ColourInformationBox with the colour\_type set to 'prof' and with ICC\_profile contents being icc\_data |
| 6 | has\_separate\_alpha\_item is true and alpha\_item\_codec\_config\_size is not 0 | Item property with the type set to codec\_config\_type and with contents from alpha\_item\_codec\_config |
| 7 | has\_separate\_alpha\_item is true | AuxiliaryTypeProperty with aux\_type set to urn:mpeg:mpegB:cicp:systems:auxiliary:alpha. |
| 8 | has\_separate\_alpha\_item is true and is\_float is 0 | PixelInformationProperty with num\_channels set to 1 and bits\_per\_channels set to bit\_depth\_minus\_one+1.  NOTE: This should be updated once we have an accepted proposal for GitHub issues #69. |
| 9 | false | Reserved |
| 10 | false | Reserved |
| 11 | false | Reserved |
| 12 | false | Reserved |
| 13 | false | Reserved |
| 14 | false | Reserved |
| 15 | false | Reserved |
| 16 | false | Reserved |

If extended\_meta contains an ItemPropertyContainerBox inside an ItemPropertiesBox its entries shall be appended to the entries described above.

The ItemPropertyAssociationBox shall have the entries below. Any association to a FreeSpaceBox shall be dropped.

* Item 1 shall be associated with ItemPropertyContainerBox entries:
  + 1, essential
  + 2, non-essential
  + 3, non-essential
  + 4, essential
  + 5, essential
* If has\_separate\_alpha\_item is true, item 2 shall be associated with ItemPropertyContainerBox entries:
  + 6, essential
  + 2, non-essential
  + 7, essential
  + 8, non-essential

If extended\_meta contains an ItemPropertyAssociationBox inside an ItemPropertiesBox, associations for items 1 and 2 shall be appended to the respective entries above, while entries for other items shall be appended as separate entries.

1. **ItemLocationBox**

The expanded MetaBox shall have an ItemLocationBox of version 1 or version 2 containing the following entries:

* item\_ID 1, with construction\_method set to 1, offset set to alpha\_item\_data\_size and length set to main\_item\_data\_size\_minus\_one+1.
* Optional item\_ID 2, with construction\_method set to 1, offset set to 0 and length set to alpha\_item\_data\_size.
* Optional item\_ID 3, with construction\_method set to 1, offset set to main\_item\_data\_size\_minus\_one+1+alpha\_item\_data\_size, and length set to exif\_data\_size\_minus\_one+1.
* Optional item\_ID 4, with construction\_method set to 1, offset set to main\_item\_data\_size\_minus\_one+1+alpha\_item\_data\_size+exif\_data\_size\_minus\_one+1, and length set to xmp\_data\_size\_minus\_one+1.

If extended\_meta contains an ItemLocationBox, its entries shall be appended to the entries above.

1. **ItemDataBox**

The expanded MetaBox shall have an ItemDataBox containing alpha\_data, main\_data, exif\_data, and xmp\_data concatenated in that order.

1. **Other boxes**

Other boxes present in the extended\_meta that are not explicitly mentioned in the sections above shall be appended as is to the expanded MetaBox.

### Alternatives considered

#### New container format

One of the main use-cases triggering this proposal was specifically for use on the web. The web community did not want a new special-purpose container format.

#### Providing default values for boxes in 'meta'

There are a number of boxes in HEIF that could be dropped if provided with default values. Examples are PrimaryItemBox, DataInformationBox and others. This was explored, but did not give enough gains.

HEIF as it is currently written is heavily optimized for many items that share properties. This is great for complicated files, but causes severe overhead for single-item files.

Many of the item properties are also written to be optimized for flexibility and extensibility rather than small size. As an example, ImageSpatialExtentsProperty is a FullBox with 32-bit integers for width and height. This means it requires 20 bytes to specify the dimensions of an item, which is a significant number of bytes if the payload itself is only 100 bytes.

The solution to this is to create a new dedicated box with variable length fields. Storing the actual payloads inside this box allows us to get rid of the 8-bytes required for a separate ItemDataBox or MediaDataBox. This also allowed us to mandate a strict payload order within the box, which is one of the features requested for the web in order to make streamed decoding better.

#### Codec-specific solutions

Various codec-specific solutions were considered, where the specification would instead document how various missing boxes in the MetaBox should be populated directly from the elementary stream. This had a number of drawbacks:

* It was not codec-agnostic and could only be used by a codec that fully specified how these boxes should be populated. A significant amount of work was needed for any new codec/spec that wanted to adopt it.
* It had a number of tricky edge-cases where certain information was ambiguous and hard to fully document in the specification.

#### uleb128 rather than sqlite\_varint for variable length integers

If uleb128 is used instead of sqlite\_varint, we would need one extra byte for any range between 128 and 240. uleb128 is more efficient for larger numbers, but for those the difference is less important. sqlite\_varint is used to aggressively optimize for minimal file-size.

### Examples

A screenshot of a computer program

Description automatically generated

Figure 4: Opaque item 8-bit sRGB HEIC

A screenshot of a computer

Description automatically generated

Figure 5: 8-bit sRGB HEIC with pre-multiplied alpha and extended\_meta

# References

1. ISO/IEC 23008-12, Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 12: Image File Format
2. ISO/IEC 14496-12, Information technology — Coding of audio-visual objects — Part 12: ISO base media file format
3. SMPTE ST 2113(2019), *Colorimetry of P3 Color Spaces*
4. MPEGGroup/Fileformat GitHub issue tracker, “*HEIF needs a reduced header mode”,* <https://github.com/MPEGGroup/FileFormat/issues/59>
5. ISO/IEC 23091-2:2019: *Information technology — Coding-independent code points — Part 2: Video*
6. Requirements for still image coding using HEVC, [w13823](https://dms.mpeg.expert/doc_end_user/current_document.php?id=45059&id_meeting=0" \t "_blank)