ISO/IEC JTC 1/SC 29/WG 03 N0994



**Document type:** Output Document

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

**Status:** Approved

**Date of document:** 2023-08-11

**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 N0994**

**July 2023 – Geneva, CH**

|  |  |
| --- | --- |
| **Title** | **Exploration on low-overhead HEIF-compatible image file format** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **22997** |

# 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 even 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 that will be discussed by the group in subsequent meetings:

* **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.

# 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.

## Condensed image item

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

### 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 Condensed Image Item Box ('coni'), 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)
* 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 'hasExtendedMeta' 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 CondensedImageBox is used in a file:

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

The 'coni' 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 'coni' brand may be used, in which case 'hasExplicitCodecTypes' 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 hasExtendedMeta is set to true and allows for adding additional image items and/or grouping image items to tracks.

### Experimental results

Below is an example image with a 40x30 P3D65 [3] payload of 300 bytes that has a codec specific 'coni' 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 ‘coni’brand)  
 - 4 bytes minor version  
 - 4 bytes compatible brands (codec specific ‘coni’brand repeated)  
coni: 17 bytes  
 - 64 bits (8 bytes) for box header  
 - 2 bits version  
 - 8 bits width  
 - 8 bits height  
 - 1 bits isFloat  
 - 4 bits bitdepth  
 - 1 bits isMonochrome  
 - 1 bits isFullRange  
 - 2 bits colourType  
 - 15 bits NCLX  
 - 1 bits hasExplicitCodecTypes  
 - 8 bits mainItemCodecConfigSize  
 - 16 bits mainItemDataSize  
 - 1 bits hasAlpha  
 - 1 bits hasExtendedMeta  
 - 1 bits hasExif  
 - 1 bits hasXMP  
 - 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

#### Condensed Image Item Box

##### Definition

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

The condensed 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 CondensedImageBox is present, a file-level MetaBox shall not be present in the file. However, the body of a MetaBox may be embedded in the CondensedImageBox when the hasExtendedMeta flag is set to one.

The major\_brand of the FileTypeBox may be specified in derived specifications to signal pre-defined values for infeType and codecConfigType. However, if no such codec specific brand exists, the 'coni' brand may be used, in which case hasExplicitCodecTypes shall be set to 1.

The CondensedImageBox may be followed by a MovieBox.

The CondensedImageBox may be followed by a MediaDataBox.

When processing the CondensedImageBox it is expanded to a full MetaBox containing the following boxes:

* HandlerBox with handler\_type equal to 'pict'.
* PrimaryItemBox with item\_ID set to 1.
* ItemInfoBox containing the following entries:
  + ItemInfoEntry with item\_ID set to 1 and item\_type set to infeType.
  + Optional ItemInfoEntry with item\_ID set to 2 and item\_type set to infeType.
  + Optional ItemInfoEntry with item\_ID set to 3 and item\_type set to Exif.
  + Optional ItemInfoEntry with item\_ID set to 4 and item\_type set to mime and content\_type set to 'application/rdf+xml'.
* ItemReferenceBox containing the following entries:
  + Optional 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.
  + Optional 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.
  + Optional item type reference with referenceType set to 'auxl', reference\_count set to 1, from\_item\_ID set to 4 and to\_item\_ID set to 1.
* ItemPropertiesBox with the following entries:
  + ItemPropertyContainerBox with the following properties:
    - Property with the type set to codecConfigType and with contents from mainItemCodecConfig
    - ImageSpatialExtentsProperty with image\_width set to width and image\_height set to height from the CondensedImageBox
    - PixelInformationProperty with the values from the CondensedImageBox:  
      num\_channels is set to 1 if isMonochrome is 1 and num\_channels is set to 3 otherwise. 1 is added to num\_channels if hasAlpha is 1 and alphaItemDataSize is 0 (meaning the codec supports native alpha channels).  
      If isFloat is 0, bits\_per\_channel is set to bitDepthMinusOne+1 for each channel. Otherwise, bits\_per\_channel is set to 16, 32, or 64 for each channel for floatPrecision respectively being 0, 1, or 2.
    - ColourInformationBox with the values from the CondensedImageBox:  
      colour\_type is set to 'nclx'.  
      If colourType is set to 0, the colour space is considered to be sRGB and the colour\_primaries, transfer\_characteristics, and matrix\_coefficients are set to 1, 13 and 6 respectively.  
      If colourType is set to 1 or 2, the colour\_primaries, transfer\_characteristics and matrix\_coefficients are set to colourPrimaries, transferCharacteristics and matrixCoefficients respectively.  
      If colourType is set to 3, the colour\_primaries and transfer\_characteristics are set to 2 (unspecified). The matrix\_coefficients is set to matrixCoefficients.
    - Either another ColourInformationBox with the colour\_type set to 'rICC' and with ICC\_profile contents being iccData, or a FreeSpaceBox if colourType is not 3.
    - Either a property with the type set to codecConfigType and with contents from alphaItemCodecConfig, or a FreeSpaceBox if either hasAlpha or alphaItemDataSize is 0.
    - Either an AuxiliaryTypeProperty with aux\_type set to urn:mpeg:mpegB:cicp:systems:auxiliary:alpha, or a FreeSpaceBox if either hasAlpha or alphaItemDataSize is 0.
    - Either a FreeSpaceBox if either hasAlpha or alphaItemDataSize is 0, or a PixelInformationProperty with the values from the CondensedImageBox for alpha:  
      num\_channels is set to 1.  
      If isFloat is 0, bits\_per\_channels is set to bitDepthMinusOne+1. Otherwise, bits\_per\_channels is set to 16, 32, or 64 for floatPrecision respectively being 0, 1, or 2.
  + ItemPropertyAssociationBox with the following entries:
    - Item 1 associated with entries 1, 2, 3, 4, 5; all with essential set to 1.
    - Optional item 2 associated with entries 2, 6, 7, 8; all with essential set to 1.
* ItemLocationBox with the following entries
  + ID 1, with construction\_method set to 1, offset set to alphaItemDataSize and length set to mainItemDataSize.
  + Optional ID 2, with construction\_method set to 1, offset set to 0 and length set to alphaItemDataSize
  + Optional ID 3, with construction\_method set to 1, offset set to mainItemDataSize+alphaItemDataSize, and length set to exifDataSize.
  + Optional ID 4, with construction\_method set to 1, offset set to mainItemDataSize+alphaItemDataSize+exifDataSize, and length set to xmpDataSize.
* ItemDataBox containing alphaData, mainData, exifData, and xmpData concatenated in that order.

When extended MetaBox is present in CondensedImageBox it may not contain:

* ItemInfoBox entries for items 1, 2, 3 and 4
* ItemLocationBox entries for items 1, 2, 3 and 4
* ItemLocationBox entries for other items using construction method 1
* HandlerBox
* DataInformationBox
* ItemDataBox
* PrimaryItemBox
* An alpha image item auxiliary to the primary item 1

If the extended MetaBox contains ItemPropertyContainerBox (inside ItemPropertiesBox), the following rules apply:

* All properties are appended to the ItemPropertyContainerBox in the extended MetaBox described above.

If the extended MetaBox contains ItemPropertyAssociationBox (inside ItemPropertiesBox), the following rules apply:

* All item property indices refer to the concatenated ItemPropertyContainerBox as synthesized from the CondensedImageBox and the extended MetaBox as described above.
* The CondensedImageBox will always create 8 item properties, so to refer to the first property in the extended MetaBox, ItemPropertyContainerBox, use index 9.
* Any entries for items 1, 2, 3, and 4 are merged into the entries in the extended MetaBox described above.

If the extended MetaBox contains ItemLocationBox, the following rules apply:

* File offsets are offsets in the real file and not in the synthesized file.

##### Syntax

aligned(8) class CondensedImageBox extends Box('coni') {  
 bit(2) version;  
 width = sqlite\_varint() + 1;  
 height = sqlite\_varint() + 1;  
  
 // Colour and bit-depth  
 bit(1) isFloat;  
 if (isFloat) {  
 bit(2) floatPrecision; // 0==half-float, 1==float, 2==double, 3==?  
 }  
 else {  
 bit(4) bitDepthMinusOne;  
 }  
 bit(1) isMonochrome;  
 if (isMonochrome == 0) {  
 bit(1) isSubsampled;  
 }  
 bit(1) fullRange;

bit(2) colourType;  
 if (colourType == 0) {  
 // sRGB colour space  
 colourPrimaries = 1;  
 transferCharacteristics = 13;  
 matrixCoefficients = 6;  
 }  
 else if (colourType == 1) {  
 bit(5) colourPrimaries;  
 bit(5) transferCharacteristics;  
 bit(5) matrixCoefficients;  
 }  
 else if (colourType == 2) {  
 bit(8) colourPrimaries;  
 bit(8) transferCharacteristics;  
 bit(8) matrixCoefficients;  
 }  
 else {  
 bit(8) matrixCoefficients;  
 iccDataSize = sqlite\_varint();  
 }  
  
 // Item metadata  
 bit(1) hasExplicitCodecTypes;  
 if (hasExplicitCodecTypes) {  
 unsigned int(32) infeType;  
 unsigned int(32) codecConfigType;  
 }  
 mainItemCodecConfigSize = sqlite\_varint();  
 mainItemDataSize = sqlite\_varint() + 1;

// Other items  
 bit(1) hasAlpha;  
 if (hasAlpha) {  
 // Alpha has the following requirements:  
 // Same dimensions, bit depth, codec as main  
 // Monochrome  
 // If hasAlpha is 1 and alpha size is 0, it means that main image codec   
 // supports interleaved alpha  
 bit(1) alphaIsPremultiplied;  
 alphaItemCodecConfigSize = sqlite\_varint();  
 alphaItemDataSize = sqlite\_varint() + 1;  
 }

bit(1) hasExtendedMeta;  
 if (hasExtendedMeta) {  
 extendedMetaSize = sqlite\_varint() + 1;   
 }  
  
 bit(1) hasExif;  
 if (hasExif) {  
 exifDataSize = sqlite\_varint() + 1;  
 }  
  
 bit(1) hasXMP;  
 if (hasXMP) {  
 xmpDataSize = sqlite\_varint() + 1;  
 }  
  
 // Pad bits until byte-aligned  
 trailing\_bits();  
  
 // Actual data  
 unsigned int(8) alphaItemCodecConfig[alphaItemCodecConfigSize]; // Codec config body data  
 unsigned int(8) mainItemCodecConfig[mainItemCodecConfigSize]; // Codec config body data  
 unsigned int(8) extendedMeta[extendedMetaSize]; // Embedded 'meta' box  
 unsigned int(8) iccData[iccDataSize]; // ICC profile data  
 unsigned int(8) alphaData[alphaItemDataSize]; // Alpha elementary stream  
 unsigned int(8) mainData[mainItemDataSize]; // Main image elementary stream  
 unsigned int(8) exifData[exifDataSize]; // Exif metadata  
 unsigned int(8) xmpData[xmpDataSize]; // XMP metadata  
}

##### Semantics

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

width: specifies the width of the reconstructed image in pixels, as specified in ImageSpatialExtentsProperty in clause 6.5.3

height: specifies the height of the reconstructed image in pixels, as specified in ImageSpatialExtentsProperty in clause 6.5.3

isFloat: specifies whether floatPrecision or bitDepthMinusOne are signalled. If isFloat is set to 1 indicates that the floatPrecision is signalled, otherwise bitDepthMinusOne is  signalled.

floatPrecision: 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 isFloat is set to 0, the value is undefined.

bitDepthMinusOne: plus 1 indicates the maximum number of bits per channel for the pixels of the reconstructed image of every associated image item.

isMonochrome: when set to 1 indicates that there is exactly one channel of coded colour samples, otherwise there are exactly three channels of coded colour samples.

isSubsampled: 0 indicates that there is the same number of samples in each colour channel. 1 indicates that the chroma planes are subsampled compared to the luma plane (4:2:0). Set to 0 if isMonochrome is 1. The meaning of the value of isSubsampled shall match the contents of the mainItemCodecConfig.

fullRange: carries a VideoFullRangeFlag value as defined in ISO/IEC 23091-2

colourType: specifies the colour encoding type. When set to 0 indicates sRGB. 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 is present.

colourPrimaries: carries a ColourPrimaries value as defined in ISO/IEC 23091-2

transferCharacteristics: carries a TransferCharacteristics value as defined in ISO/IEC 23091-2

matrixCoefficients: carries a MatrixCoefficients value as defined in ISO/IEC 23091-2

iccDataSize: specifies the size of ICC profile data when the colourType field indicates it is present in bytes. Set to 0 unless colourType is 3.

hasExplicitCodecTypes: when set to 1 indicates that both infeType and codecConfigType are explicitly signalled, otherwise their types are implied from the major\_brand of the FileTypeBox.

infeType: corresponds to the item\_type field of the version 2 of the ItemInfoEntry box. Defined by the major brand if hasExplicitCodecTypes is set to 0.

codecConfigType: corresponds to the codec configuration box type. Defined by the major brand if hasExplicitCodecTypes is set to 0.

mainItemCodecConfigSize: specifies the size of the configuration for the main image item.

mainItemDataSize: specifies the size of the data for the main image item in bytes.

hasAlpha: 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.

alphaIsPremultiplied: when set to 1 indicates that alpha values are pre-multiplied, otherwise alpha values are not pre-multiplied.

alphaItemCodecConfigSize: 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 hasAlpha is 0. The value shall be 0 if alphaItemDataSize is 0.

alphaItemDataSize: specifies the size of the data for the alpha image item in bytes. If hasAlpha 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 mainData chunk. Shall be set to 0 if hasAlpha is 0.

hasExtendedMeta: when set to 1 indicates the presence of an extended MetaBox within the CondensedImageBox, otherwise it indicates the absence of it.

extendedMetaSize: specifies the size of the extended metadata in bytes.

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

exifDataSize: specifies the size of the Exif metadata in bytes.

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

xmpDataSize: specifies the size of the XMP metadata in bytes.

trailing\_bits: shall be 0.

alphaItemCodecConfig: specifies the optional alpha image codec configuration data. When hasAlpha is set to 0 alphaItemCodecConfig is not present.

mainItemCodecConfig: specifies the main image item codec configuration data.

extendedMeta: specifies the optional extended metadata. When hasExtendedMeta is set to 0, extendedMeta is not present.

iccData: specifies the optional ICC profile data. When colourType is not set to 3 iccData is not present.

alphaData: specifies the optional alpha image data. When hasAlpha is set to 0 alphaData is not present.

mainData: specifies the main image data.

exifData: specifies the optional Exif metadata. When hasExif is set to 0 exifData is not present.

xmpData: specifies the optional XMP metadata. When hasXMP is set to 0 xmpData is not present.

sqlite\_varint() function reads a varint coded according to https://www.sqlite.org/src4/doc/trunk/www/varint.wiki

# 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*