ISO/IEC JTC 1/SC 29/WG 03 N0965

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

**Document type:** Output Document

**Title:** Technologies under Consideration for ISO/IEC 23001-17 Carriage of uncompressed video and images in ISOBMFF

**Status:** Approved

**Date of document:** 2023-07-21

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

**No. of pages:** 1 (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 N0965**

**July 2023, Virtual**

|  |  |
| --- | --- |
| **Title** | **Technologies under Consideration for ISO/IEC 23001-17 Carriage of uncompressed video and images in ISOBMFF** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **22959 (MPEG number, digits only)** |

**Abstract**

The document contains following technologies under consideration for the ISO base media file format (ISO/IEC 23001-17):

[1. Data-agnostic compression of image and sample data 3](#_Toc140826150)

# Data-agnostic compression of image and sample data

With 23001-17 (Carriage of uncompressed video and images in ISO Base Media File Format) nearing completion, we are looking at potential improvements beyond simple uncompressed imagery. While 23001-17 nominally defines the mechanism to carry uncompressed imagery and raster data within ISOBMFF, it also more generically defines a mechanism to define the in-memory layout for an image item or sample, and then store the image using that layout:

* Component value types and sizes
* Tiling to enable efficient spatial-based access for large imagery
* Padding to ensure that individual component values or groups of component values can be accessed by the processing unit without having to perform bit shifts for every single access, and without having to cross key storage/memory page boundaries

This capability brings ISOBMFF on par with other generic storage formats for numeric data (such as HDF5) with one exception – data-agnostic numerically-lossless and bitwise-lossless compression, transparent to the end user. In HDF5, for example, a large N-dimensional array of numerical data can be chunked (i.e., tiled) and then each chunk is compressed using off-the-shelf ubiquitous data-agnostic compression tools (e.g., deflate). This provides storage and transmission savings similar to numerically lossless image coders, with minimal computational performance impact. These capabilities can be applied not only to typical integer-based pixel formats, but also to IEEE 754 floating-point pixel formats that are unsupported in most imagery compression algorithms.

Adding this capability to 23001-17 is expected to provide cost savings (for both storage and network transmission), particularly for applications and datasets involving large amounts of uncompressed content, such as geospatial and scientific imagery, without significantly changing how those applications access the pixel data.

While not necessarily required, a mechanism that could be applied to other forms of data (e.g., audio or KLV formatted metadata), this capability is not necessarily required.

## Use cases

* Data producer generates a large image (or image sequence) using 32-bit IEEE 754 binary floating point component values after calibration. The image is tiled using 1024 × 1024 tiles and each tile is independently compressed using deflate.
* Data consumer desires to load only a specific spatial region from an image or sample based on some form of chunking of the image (chunk by rectangular tiles or rows). Consumer uses offset/size information provided within the ISOBMFF file to locate only the desired chunks. Each of those chunks can be independently decompressed.
* Data consumer desires to load only a spatial region from a large tiled image, where the desired region is smaller than a region contained within a single compressed chunk. After decompressing the chunk, the order, alignment and padding of the component data is maintained, enabling the consumer to calculate individual component value offsets – parsing through the decompressed chunk to locate specific pixels/component is not necessary.
* A data producer collects an image of the Earth scanning diagonally from northwest to southeast. For simpler human viewing of the image, the collected image is rotated 45 degrees to align north to up, and additional padding is added to form a tiled, rectangular image buffer. Since the four corner tiles are entirely fill data, they are omitted from the data stored within the ISOBMFF file.
* Legal requirements for records management require bitwise-lossless compression of the image data – it is not sufficient that the decompressed pixel value is numerically equal to the original pixel value; the specific bit patterns must also be the same.

## Requirements

* Numerically and bitwise lossless compression of raw image items and track samples, especially when consisting of floating-point formatted media.
* Pixel organization prior to compression (as well as post-decompression) mirrors the uncompressed spec in 23001-17. The point is for the encoder to determine how to best organize the pixel/component data when in a directly-accessible form, and then to provide simple off-the-shelf, numerically/bitwise-lossless compression for that data.
* Utilization of existing compression technology with open licensing, broad/mature support, and availability of open-source software and tools
  + - * + Ability to access portions of the image without fully decompressing the entire sample or image item.

Ability to code and access tiles independently. This includes gridded items as well as tiles defined within 23001-17.

Investigate the ability for “unnecessary” tiles to be omitted from the data

e.g., a large geospatial image is rotated within the rectangular pixel boundaries so north is up. This causes the corners of the expanded rectangular image to be just fill pixels, with a resulting preference to not have to store those fill pixels

Specifying a transformative property for arbitrary rotation is ultimately not sufficient as there are many means to precisely georeference an image (e.g., orthorectification) involving pushing pixels around based on imaging geometry and terrain.

Determine if sub images of a gridded image item can be omitted.

* + - * + Orthogonal capability to the 23001-17 component value alignment, padding and component value organization is highly desired. Upon decompression, each chunk is used exactly as if the respective uncompressed values had been loaded directly from storage or the network.
        + Constructive interaction between transformational properties is desired. For example, if I compress the sample as individual chunks, but then encrypt the entire sample, the independence of those chunks might be lost. We note that this might not be generically possible without defining tiling-aware encryption/protection mechanism, and that might look a lot like a gridded image item.

## Suggested Implementation Approach

## Storage in media tracks

Uncompressed video tracks compliant to this specification are video tracks compliant to ISO/IEC 14496-12 that use a video sample entry with codingname equal to 'agcV', hereafter called agnostically-compressed video sample entry.

The agnostically-compressed video sample entry shall contain one one CompressionConfigBox, specifying

* one CompressionConfigBox, as defined below, specifying the specific data compression method and the “unit” of sample data (“subsample” on which the compression is applied)
* one UncompressedFrameConfigBox as defined in ISO/IEC 23001-17. This box defines the organization of the component data after the coded data as been uncompressed using the specified coding type
* one ComponentDefinitionBox, as defined in ISO/IEC 23001-17, if the UncompressedFrameConfigBox version is not 1.
* One or more SimplifiedSubSampleInformationBoxes as required to specify the location and size of each independently compressed element within the sample, as defined in this document.

The compressorname field of an uncompressed video sample entry should be set to all 0 (empty string). The depth field of an uncompressed video sample entry shall be ignored and should be set to 0, the bit depth per component being indicated by the UncompressedFrameConfigBox .

The handler type associated with the track is usually 'vide', 'auxv' or 'pict' but derived specifications may introduce new handler types. The width and height fields of the sample entry shall document the exact frame dimension, in pixels, of any sample of the video stream that is described by this sample entry. Consequently, if the frame dimension changes within a video track, multiple sample entries shall be used.

The payload of an agnostically-compressed video media sample consists of one agnostically-compressed frame.

Each agnostically-compressed video media sample is a sync sample. The SyncSampleBox, ShadowSyncSampleBox, CompositionOffsetBox and CompositionToDecodeBox shall not be present in the track.

Media tracks containing only non-visual components should be marked as not present in the presentation, i.e. track\_in\_movie flag should not be set.

Media tracks containing per-pixel meta-data components describing pixels in another track should use a track reference of type 'cdsc' to the track they describe.

## Storage in image items

An agnostically-compressed image compliant to this specification is an image item compliant to ISO/IEC 23008-12 with the item\_type 'agci'.

An agnostically-compressed image shall be associated with:

* a CompressionConfigBox, as specified above specifying how the image is compressed and the “unit” of the image data (“subsample”) on which the compression is applied.
* an UncompressedFrameConfigBox essential item property, as defined in ISO/IEC 23001-17 (i.e., essential shall be equal to 1 for an UncompressedFrameConfigBox item property associated with an image item of type 'agci'),
* a ComponentDefinitionBox essential item property, as defined in ISO/IEC 23001-17 if the UncompressedFrameConfigBox version is not 1,
* an ImageSpatialExtentsProperty whose image\_width and image\_height fields shall document the exact frame dimension, in pixels, of the reconstructed image, i.e. the size of the image before applying any associated transformative properties.
* One or more SimplifiedSubSampleInformationBoxes as required to specify the location and size of each independently compressed element within the sample, as defined in this document.

The payload of an agnostically-compressed image consists of one agnostically-compressed frame.

An agnostically-compressed image may be associated with meta-data information using the various tools defined in ISO/IEC 14496-12 or ISO/IEC 23008-12, such as descriptive item properties.

Agnostically-compressed images containing only non-visual components should be marked as hidden items, i.e. have (flags & 1) equal to 1 in their ItemInfoEntry.

agnostically-compressed images containing per-pixel meta-data components describing pixels in another item should use an item reference of type 'cdsc' to the item they describe.

### Decompression of subsamples

In order to decompress the sample or item, the decoder extracts the desired samples from the file, as specified by the SimplifiedSubSampleInformationBox. That extracted compressed data is then decompressed according to the compression algorithm specified by the compression\_type field in the CompressionConfigBox. The resultant data is exactly the data as was specified by that element of the sample or image as specified by the UncompressedFrameConfigBox, including any padding placed at the end of that element in order to align the next element. Specifics on the uncompressed bytes associated with each sample are specified by the definition of each value for the subsample\_type field in the CompressionConfigBox.

### CompressionConfigBox

class CompressionConfigBox extends FullBox('cmpC', 0, 0) {

unsigned int(32) compression\_type;

unsigned int(1) can\_decompress\_full\_sample;

unsigned int(7) subsample\_type;

}

with compression\_type: 4CC indicating the compression mode for the sample or image item

* 'defl': deflate algo
* ‘brot’: brotli compression
* ...

The definition of each value of compress\_type specifies not only the algorithm but also the codestream format for each compressed subsample. For example, ‘defl’ specifies the use of the deflate algorithm as packaged in the zlib format defined by IETF RFC 1950.

can\_decompress\_full\_sample: specifies whether the entire sample, item or data can be decompressed as a single large input buffer (value 1), or whether each subsample must be decompressed individually (value 0). If the value is 1 and the decoder chooses to decompress the entire sample, item or data, then the resultant decompressed data shall be equal to the concatenation of each decompressed subsample, with zero bytes of padding or fill inserted between the decompressed subsamples.

subsample\_type: indicate the entity being compressed within an agnostically-compressed video sample or image item. Each subsample contains exactly the result of the compression algorithm applied to the identified entity.

* 0: the entity is the full object (e.g., sample, item)
* 1: the entity is the full image for a given component (component-based interleave and mixed interleave)
* 2: the entity is a tile
* 3: the entity is a row
* 4: the entity is a pixel
* other modes to be discussed depending on interleaving constrained.
* If this capability can be applied to other types of data, consider making the values dependent on the type of item (e.g., the above values for video samples and image items, but for KLV tracks the values might be keys).

If subsample\_type is 0, then can\_decompress\_full\_sample shall be 1.

### SimplifiedSubSampleInformationBox

The SimplifiedSubSampleInformationBox provides information on the location and size of each subsample within the agnostically-compressed element, where the type of each subsample is specified by the subsample\_type field in the CompressionConfigBox. An initial suggestion is to use the SubSampleInformationBox itself, although that box may introduce unnecessary complexity and significant semantic differences, causing developer confusion and creating excessive possibility for invalid data.

Ed Note: If use of SubSampleInformationBox is strongly desired, please provide recommendations for profiling it down to just what is needed for this use.

A possible structure of the SimplifiedSampleInformationBox is as follows:

class SimplfiedSampleInformationBox extends FullBox('smsi', version=0, flags) {

unsigned int(32) num\_subsamples;

unsigned int size = ((flags & 1) + 1) \* 16; // 16 or 32

unsigned int contiguous = flags & 2;

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

if (!contiguous) {

unsigned int(size) subsample\_offset;

}

unsigned int(size) subsample\_size;

}

}

num\_subsample is the number of subsamples specified in the box. The value must be equal to the number of the elements specified by subsample\_type field in the CompressionConfigBox. (e.g., if subsamples are tiles, then this must be equal to the number of tiles in the sample).

subsample\_offset is the offset in bytes to the first byte of the compressed subsample. If the contiguous flag is set, then all subsamples are stored within a contiguous range of bytes within the sample or item, and the first byte of the first subsample is at the first byte of the sample (e.g., as specified by iloc).

subsample\_size is size in bytes of this subsample. The value must be equal to the number of the elements specified by subsample\_type field in the CompressionConfigBox. (e.g., if subsamples are tiles, then this must be equal to the number of tiles in the sample). If the value of subsample\_size is 0, then this subsample is omitted from the file and the decoder should populate that portion of the sample with default values; e.g., display that portion of the image with the background color or a user-defined fill color.

Ed Note: This is not the typical convention, where a size of zero means all the bytes until the end of the container/file. It is not believed that allowing the producer to omit the actual size will detrimentally affect implementations for this case. However, if the committee feels that we must stick to that convention, we can find another solution (e.g., set to MAXINT for the respective size data type). Alternately, the producer can create a single subsample with default values and store it once in the file. Any “omitted” subsamples can point to that same block of data.

## Potential improvements

The following potential improvements can be under consideration if enough support is shown and value can be demonstrated

* Allowing different subsamples to be compressed differently
* Allowing subsamples to be stored externally to the file defining the sample or item
* Allowing a mix of subsamples in the file and in external files