ISO/IEC JTC 1/SC 29/WG 03 N0308

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

**Document type:** Output Document

**Title:** Exploration of ISO/IEC 23001-17 Carriage of Uncompressed Video in ISOBMFF

**Status:** Approved

**Date of document:** 2021-08-02

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

**No. of pages:** 12 (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 N0308**

**August 2021, Virtual**

|  |  |
| --- | --- |
| **Title** | **Exploration of ISO/IEC 23001-17 Carriage of Uncompressed Video in ISOBMFF** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **20562** |

1. Introduction

At MPEG#122, Telecom ParisTech presented an input contribution on uncompressed audio and video storage in ISOBMFF where several existing practices for storage were summarized and several missing features in ISOBMFF were identified [1]. While the audio aspects have been standardized as ISO/IEC 23003-5, the carriage of uncompressed video in ISOBMFF has not been resolved. It was pointed out that the current state of uncompressed video carriage in ISOBMFF has a lack of documentation and features. Furthermore, the carriage of uncompressed video should reuse as many existing tools as possible, but it is unclear what the state of the various specifications and formats is today.

At MPEG#132, Telecom ParisTech presented a second contribution on this topic, introducing a possible approach for carriage of uncompressed video [2], while Qualcomm presented a set of requirements for uncompressed video carriage based on the related activity in 3GPP [3]. The File Format group has agreed that there is a need for unified solution for carriage of uncompressed video in ISOBMFF in the industry and requested a subdivision of ISO/IEC 23001 into a next part 17.

At MPEG#133, Canon Research Centre France presented further use cases and requirements for the carriage of uncompressed video in ISOBMFF based on industrial cameras [4] and the group has agreed to start the exploration activity on 23001-17. Furthermore, at MPEG#135 Canon Research Centre France and Telecom Paris presented a joint contribution discussing the limitations of a 4CC-based sample entry descriptions for uncompressed video formats, especially regarding the support of a great number of existing formats as well as the extensibility to new formats [13].

This document can be considered as the starting point of the exploration on 23001-17, in which we try to collect best industry practices for carriage of uncompressed video, identify a set of features / requirements which should be addressed in the new MPEG specification, and identify any possible overlaps and gaps with current implementations.

1. Existing code points

In this section we investigate several existing methods for uncompressed video carriage. Please note that the term *uncompressed*, which is used throughout this document, is slightly inaccurate as chroma sub-sampling is also a form of data compression.

* 1. Technical Note TN2162

Technical Note [TN2162](https://developer.apple.com/library/archive/technotes/tn2162/_index.html) is originally known as Ice Floe Dispatch 19 is a part of the “Letters from the Ice Floe” from the QuickTime Engineering team which was originally published on December 14th, 1999 [5]. This document is no longer being updated and is stored in the Apple Documentation Archive, but it can be considered as being de facto standard for carriage of uncompressed video in QTFF [6]. Furthermore, several code points defined in TN2162 (‘2vuy’, ‘v210’) are now used in various implementations. For example, Final Cut Pro allows the users to extract the uncompressed version of the master file in two different formats packaged in a QTFF (.mov) file:

* *Uncompressed 8-bit 4:2:2*: which is using the ‘2vuy’ FOURCC
* *Uncompressed 10-bit 4:2:2*: which is using the ‘v210’ FOURCC

A list of Y’CbCr packing types as defined in TN2162 is listed below:

Table 1: Compression types for storing uncompressed Y’CbCr data

|  |  |  |
| --- | --- | --- |
| Compression Type | FourCC | Description |
| k422YpCbCr8CodecType | '2vuy' | 8-bit-per-component 4:2:2 |
| kComponentVideoCodecType | 'yuv2' | 8-bit-per-component 4:2:2 |
| k444YpCbCr8CodecType | 'v308' | 8-bit-per-component 4:4:4 |
| k4444YpCbCrA8CodecType | 'v408' | 8-bit-per-component 4:4:4:4 |
| k422YpCbCr16CodecType | 'v216' | 10,12,14,16-bit-per-component 4:2:2 |
| k444YpCbCr10CodecType | 'v410' | 10-bit-per-component 4:4:4 |
| k422YpCbCr10CodecType | 'v210' | 10-bit-per-component 4:2:2 |

Detailed information on the individual compression types from Table 1 can be found in [TN2162](https://developer.apple.com/library/archive/technotes/tn2162/_index.html). As we will show later, ‘2vuy’ and ‘v210’ are the most supported ones today.

* 1. SMPTE

Several SMPTE specifications also define the transport and storage of uncompressed video.

* + 1. ST 2110

SMPTE ST 2110 is a suite of specifications that defines the RTP-based transport of digital media over IP networks. ST 2110 is mainly intended to be used by broadcasters in live production workflows where quality of the video is more important than the required bandwidth [7]. SDI to IP converters are typically used to provide a transition from SDI to all-IP networks [8].

Part 20 of the ST 2110 defines RTP-based transport of uncompressed video, which specifies the format in which samples of uncompressed video data are sent over IP [9]. In addition, it defines a set of required media type properties which are signaled using the Session Description Protocol including:

* Sampling (color difference sub-sampling structure)
  + YCbCr: 4:4:4 / 4:2:2 / 4:2:0 (BT.601, BT.709, BT.2020, BT.2100)
  + CLYCbCr: 4:4:4 / 4:2:2 / 4:2:0 (BT.2020)
  + ICtCp: 4:4:4 / 4:2:2 / 4:2:0 (BT. 2100)
  + RGB
  + XYZ (Defined in SMPTE ST 428-1)
* Depth
  + 8
  + 10
  + 12
  + 16
  + 16f (as defined in SMPTE ST 2065-1 and ITU-R BT.2100)
* Width and height
* Framerate
* Colorimetry
  + BT601, BT709, BT2020, BT2100
  + ST2065-1, ST2065-3
* Packing mode
* Interlacing
* Transfer characteristics
  + SDR, PQ, HLG, LINEAR, BT2100LINPQ, BT2100LINHLG, ST2065-1, ST428-1, DENSITY
* Range
* Pixel aspect ratio
  + 1. Material Exchange Format

SMPTE also defines the ST 377-1 “Material Exchange Format (MXF)” [10], a File Format Specification which specifies the data structure for network transport and storage of multimedia data. Mapping of uncompressed pictures into the generic container of MXF are further specified in ST 384M [11]. Annex G of [10] provides further details on picture descriptor properties used in MXF including:

* Frame layout (progressive vs interlaced [4 types of interlacing])
* Sample rate
* Aspect Ratio
* Width and height
* Field dominance (Frame dominance in temporal order of interlaced frame)
* Alpha Transparency
* Transfer characteristic
* Image alignment offset (byte alignment of units)
* Image start/end offset (skip bytes)
* Picture Essence Coding (concept similar to FOURCC), other properties can be used in combination.
* Component bit depth
* Horizontal / Vertical Subsampling
* Color Siting (used to compute subsampled color difference values)
* Alpha sample depth
* Black / White reference level (luma value for black / white reference)
* Color range
* Pixel layout for RGBA only
* Palette for RGBA (reference colors by indexes)
* Scanning direction (8 modes lrtb, lrbt, lrbt, rlbt, tblr, tbrl, btlr, btrl)
  1. MP4RA.org

There are also several ISOBMFF code points already registered at [MP4RA.org](http://mp4ra.org/) that are listed in Table 2 below. However, the linked specification [Digital Voodoo](http://www.digitalvoodoo.net/) is no longer available, so it is not really clear what exactly these 5 code points represent.

Table 2: Uncompressed video code points at mp4ra.org

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Description | Handler | Specification |
| DVOO | Digital Voodoo 8 bit Uncompressed 4:2:2 codec | [Video](http://mp4ra.org/handler.html#vide) | [Digital Voodoo](http://mp4ra.org/specs.html#voodoo) |
| DVOR | Digital Voodoo intermediate raw | [Video](http://mp4ra.org/handler.html#vide) | [Digital Voodoo](http://mp4ra.org/specs.html#voodoo) |
| DVTV | Digital Voodoo intermediate 2vuy | [Video](http://mp4ra.org/handler.html#vide) | [Digital Voodoo](http://mp4ra.org/specs.html#voodoo) |
| DVVT | Digital Voodoo intermediate v210 | [Video](http://mp4ra.org/handler.html#vide) | [Digital Voodoo](http://mp4ra.org/specs.html#voodoo) |
| HD10 | Digital Voodoo 10 bit Uncompressed 4:2:2 HD codec | [Video](http://mp4ra.org/handler.html#vide) | [Digital Voodoo](http://mp4ra.org/specs.html#voodoo) |

Digital Voodoo appears to be an Australian video card manufacturer for professional workflows, but it seems that they are no longer in business as their website is down and their video boards are listed as deprecated on several resources such as [Viz Engine 4.0](https://documentation.vizrt.com/viz-engine-guide/4.0/Deprecated_Video_Boards.html). Nevertheless, some information on the above FOURCC’s could be found:

* ‘DVOO’: is implemented in FFmpeg as AV\_PIX\_FMT\_YUYV422 which is a packed YUV 4:2:2, 16bpp, Y0 Cb Y1 Cr, in addition DVOO is also mentioned [here](https://wiki.multimedia.cx/index.php?title=QuickTime_container).
* ‘DVOR’: according to several [websites](https://abcavi.kibi.ru/fourcc.php?fcc=DVOR&title=DVOR+-+BlueFish444+%28lossless+RGBA%2C+YUV+10-bit%29) is most likely a 16-to-10-bit RGBA
* ‘HD10’: is most likely a 10-bit YUVA (YUV + Alpha)

YUV4MPEG

YUV4MPEG is a stream format for uncompressed YUV data which defines a very simple syntax elements to describe the YUV data. The second version of the format called YUV4MPEG2 has a very confusing name and should not be mistaken for MPEG-2 transport or program streams. It is not entirely clear where the initial version of the YUV4MPEG format was first defined, but it looks like it was first introduced in the [MJPEG Tools](https://mjpeg.sourceforge.io/) project, which in itself is a set of video and audio processing tools on Linux. Currently, tools like FFMpeg and various YUV players implement support for YUV4MPEG2 files.

Files formatted in YUV4MPEG2 format use the file extension “.y4m” and begin with exactly one ‘\n’ terminated STREAM-HEADER followed by an unlimited number of FRAMEs. The STREAM-HEADER starts with the magic string “YUV4MPEG2” followed by unlimited number of TAGGED-FIELDs each preceded by a single space separator. The STREAM-HEADER ends with a single ‘\n’ line terminator.

Each FRAME consists of one ‘\n’ terminated FRAME-HEADER followed by the actual YUV data for that frame. Each FRAME-HEADER starts with the magic string “FRAME” followed by unlimited number of TAGGED-FIELDs each preceded by a single space separator.

Each TAGGED-FIELD consists of a single ascii character tag and a VALUE which does not contain whitespace. Each VALUE is either an integer (base 10 ascii representation), or RATIO (numerator:denominator), or a single ascii character, or non-whitespace ascii string.

The following tags are supported in a STREAM-HEADER:

* W[integer] - frame width in pixels, must be > 0 (required)
* H[integer] - frame height in pixels, must be > 0 (required)
* C[string] - chroma subsampling, image data format (has default)
  + 420jpeg - 4:2:0 with JPEG/MPEG-1 siting (default)
  + 420mpeg2 - 4:2:0 with MPEG-2 siting
  + 420paldv - 4:2:0 with PAL-DV siting
  + 411 - 4:1:1
  + 422 - 4:2:2
  + 444 - 4:4:4 (no subsampling)
  + 444alpha - 4:4:4 with an alpha channel
  + mono - luma (Y') plane only
* I[char] - interlacing specification: (has default)
  + ? - unknown (default)
  + p - progressive/none
  + t - top-field-first
  + b - bottom-field-first
  + m - mixed-mode: refer to 'I' tag in frame header
* F[ratio] - frame-rate (has default of 0:0 == unknown)
* A[ratio] - sample aspect ratio (has default of 0:0 == unknown)
* X[string] - 'metadata' (optional; unparsed, but passed around)

The following tags are supported in a FRAME-HEADER:

* I[string] - framing and sampling (required if-and-only-if “Im” is present in STREAM-HEADER). Value is a string of three characters "xyz" which have the following meanings:
  + x: frame presentation
    - t - top-field-first
    - T - top-field-first and repeat
    - b - bottom-field-first
    - B - bottom-field-first and repeat
    - 1 - single progressive frame
    - 2 - double progressive frame (repeat)
    - 3 - triple progressive frame (repeat)
  + y: frame temporal sampling
    - p - progressive (fields sampled at same time)
    - i - interlaced (fields sampled at different times)
  + z: frame chroma-subsampling
    - p - progressive (subsampling over whole frame)
    - i - interlaced (each field subsampled independently)
    - ? - unknown (allowed only for non-4:2:0 subsampling)
* X[string] - 'metadata' (optional; unparsed, but passed around)
  1. Formats

Many different format types are defined in a number of different places, here is a list of some of them:

* Microsoft
  + [Uncompressed RGB Formats](https://docs.microsoft.com/en-us/windows/win32/medfound/video-subtype-guids#uncompressed-rgb-formats)
  + [Recommended 8-bit YUV Formats](https://docs.microsoft.com/en-us/windows/win32/medfound/recommended-8-bit-yuv-formats-for-video-rendering)
  + [10-bit and 16-bit YUV Formats](https://docs.microsoft.com/en-us/windows/win32/medfound/10-bit-and-16-bit-yuv-video-formats)
* Apple
  + Core Media [Pixel Format Constants](https://developer.apple.com/documentation/coremedia/cmformatdescription/1564244-video_pixel_format_constants)
  + Core Video [Pixel Format Identifiers](https://developer.apple.com/documentation/corevideo/1563591-pixel_format_identifiers)
* GPAC
  + GF\_PixelFormat: [Supported pixel formats for video](https://doxygen.gpac.io/group__cst__grp.html#gaba58a85c71145a5a469688a6223d7f82)
* VideoLAN
  + [Documentation on YUV formats](https://wiki.videolan.org/YUV)
* FFmpeg
  + [AVPixelFormat](https://ffmpeg.org/doxygen/trunk/pixfmt_8h.html#a9a8e335cf3be472042bc9f0cf80cd4c5)
  + [AVCodecID](https://ffmpeg.org/doxygen/trunk/group__lavc__core.html#gaadca229ad2c20e060a14fec08a5cc7ce)
* GStreamer
  + [Raw Video Media Types](https://gstreamer.freedesktop.org/documentation/additional/design/mediatype-video-raw.html?gi-language=c): video/x-raw
* Fourcc.org
  + [YUV Pixel Formats](https://www.fourcc.org/yuv.php)
  + [RGB Pixel Formats](https://www.fourcc.org/rgb.php)
* Simple uncompressed video format for astronomical capturing [12]

From the above list it can be seen that the amount of different formats and associated FOURCCs currently implemented in various software is very large, and this fact should be taken into account when specifying the uncompressed carriage in ISOBMFF.

1. Requirements

Taking into account all 5 input contributions on this topic presented during last MPEG meetings, we construct a set of requirements [1][2][3][4]. When storing the uncompressed video format in ISOBMFF, signaling for the following attributes should be considered:

* Width and height
* Framerate and duration
* Chroma format
* Chroma Sub-sampling
* Picture aspect ratio
* Pixel aspect ratio
* Bit depth (chroma and luma for YUV, uniform for RGB)
* Color space format (YUV, RGB)
* Transfer characteristics
* YUV formats with side channel alpha / depth components
  + Note that the depth field of VisualSampleEntry in ISOBMFF is fixed to 0x0018 – images are in colour with no alpha.
* Field ordering signaling for interlaced content
* Support for several different pixel organizations, packing modes: planar, packed, semi-planar NV12/21 (hybrid planar Y + packed UV)
* Signaling for packed ordering (YUYV / YVYU / UYVY / VYUY, ARGB / ABGR / RGBA / BGRA, etc.)
* Define a sample format on how the data is stored in samples (interlacing, endianness)
* Take CICP metadata into account
* Embedded and using the data reference
* Composition of the images of the video in spatial zones with associated specific orders of groups of pixels (Figure 1)
* Support images consisting in “packed regions” (Figure 2)
* Access to spatial areas in a particular frame of the video (Figure 3)
* Other informative metadata e.g.: [3GPP raw schema](https://github.com/haudiobe/5G-Video-Content/blob/main/3gpp-raw-schema.json)

Graphical user interface

Description automatically generated

Figure 1: Zone and scan order in some industrial cameras where pixel data is available on a single tap. Pixel packets from different zones can be arranged in different orders [4].

|  |  |
| --- | --- |
| Application  Description automatically generated with medium confidence  Figure 2: Image as “packed regions” [4] | A picture containing box and whisker chart  Description automatically generated  Figure 3: Image with regions of interest [4] |

Another important question that needs to be answered was raised during MPEG#135 while discussing the input contribution from Canon Research Centre France and Telecom Paris [13]. It concerns the limitations of a 4CC-based sample entry descriptions for uncompressed video formats, in particular with respect to the support of a large number of existing formats as well as the future extensibility to new formats.

There is a huge variety of uncompressed video formats out there as discussed in section 2.5 and in the input contribution [13]. In addition, there is no registration authority for this wide variety of uncompressed video formats (excluding obsolete formats as discussed in section 2.3) and therefore some formats found in the public are identical but with different names or identification codes. At the same time some different formats may use the same identification code.

To address this problem the proponents from [13] suggested to study a generic sample entry description approach for uncompressed video which covers a wide variety of uncompressed video formats allowing representing of less common formats without having to register a separate code point for each of them. The proposed generic configuration record for uncompressed video is providing a set of configuration structures, one for each component. Where each component can be described based on its format, endianness, packing information, sampling rate, stride information etc. Furthermore, a set of additional boxes was proposed in [13] to signal chroma location (based on CICP), interlacing (based on QTFF [6]), depth information (considering disparity map and depth map) and frame packing information (based on CICP).

During the discussion at MPEG#135, the group agreed that a wild propagation of fourCCs would not be helpful, but at the same time, a single fourCC with a set of highly adjustable parameters and a confusing set of possible combinations of them would not be helpful either. The concern was that a system cannot be expected to support all possible combinations, and so neither of the two extremes is very useful.

Ideally, having a single fourCC per format would be very useful, as a system could easily detect whether such a format can be played or not without having to compare a number of configuration properties to find out. However, it seems that this will be very difficult to achieve given the huge variety formats available. However, it should be investigated whether such a small set of useful and widely used formats can be determined. If this is not possible, it should be investigated whether it is possible to divide the entire selection space into large groups with a relatively small number of fourCCs, and what criteria could be used to form such groups. Then, for each group, a set of profiles could be defined to limit the possible parameters. All these profiles should be aligned across different fourCC groups and it would be possible to define an ‘unrestricted’ profile which would allow all possible combinations for a certain group, if desired.

It was also noted by Aerospace Corporation US that for motion imagery in the Geographics Information Systems, Intelligence and Defense areas, it is not feasible to use a single fourCC per format, as a variety of different formats are used in such applications. For such applications, it is important to have a set of parameters to identify each pixel location in the data stream and to organize the pixels into different layouts so that a reader can access only a portion of the data required for a particular application (e.g. accessing data from a single band/channel from a multi-spectral sensor). In addition, tiling concepts are also important for very large imagery.

The group also noted that for the representation of depth data, defining a depth transfer functions which would allow exact mapping of depth pixel data to a depth value would be very useful.

1. Analysis of current implementations

In this section we investigate several methods on how to create ISOBMFF/QTFF files with uncompressed video data and what software is capable of playing them.

* 1. Technical tools

**FFmpeg**

FFmpeg is often referred to as a swiss army knife for multimedia processing. It can be used to generate a set of uncompressed video bitstreams as specified in [AVPixelFormat](https://ffmpeg.org/doxygen/trunk/pixfmt_8h.html#a9a8e335cf3be472042bc9f0cf80cd4c5). An example on how to generate a set of uncompressed video bitstreams can be found [here](https://github.com/podborski/isobmff/blob/raw2mov/createRaws.py). FFmpeg can even directly be used to package uncompressed video bitstreams into a .mov file. The example below packages ‘v210’ uncompressed bitstream into .mov using ‘v210’ sample entry:

ffmpeg -f v210 -s 176x144 -i gst\_v210\_176x144.raw -c copy v210\_gst.mov

**GPAC**

According to the release notes GPAC supports multiplexing of uncompressed video since version 1.0. Multiplexing uncompressed video data based on [TN2162](https://developer.apple.com/library/archive/technotes/tn2162/_index.html) seems to be considered by the developers.

**GStreamer**

A pipeline-based multimedia framework which can be used to generate uncompressed video formats as described in video/x-raw module: [Raw Video Media Types](https://gstreamer.freedesktop.org/documentation/additional/design/mediatype-video-raw.html?gi-language=c). An example on how to generate uncompressed video bitstreams can be found [here](https://github.com/podborski/isobmff/blob/raw2mov/createRaws.py).

**Final Cut Pro**

Video editing software capable of exporting and importing uncompressed ‘2vuy’ and ‘v210’ video to QTFF.

**HDRTools**

HDRTools is a software package which is used to support video standardization work in several groups e.g. JVET. It is highly configurable and already supports a wide range of uncompressed video formats. HDRTools does not currently support ISOBMFF multiplexing, but can be extended to provide such functionality. For the multiplexing part, ISOBMFF reference software ([libisomedia](https://github.com/MPEGGroup/isobmff)) can be used in a wrapper module of the HDRLib. This approach is currently implemented in a separate branch [feature\_mp4](https://gitlab.com/standards/HDRTools/-/tree/feature_mp4) and allows to create .mp4 files with the ‘v210’ format as defined in [TN2162](https://developer.apple.com/library/archive/technotes/tn2162/_index.html). The use of HDRTools in combination with libisomedia should be considered by the group while working on ISO/IEC 23001-17.

* 1. Playback of uncompressed files

To quickly test support for playback of various uncompressed sample entry types, we used a simple tool called [raw2mov](https://github.com/podborski/isobmff/tree/raw2mov/IsoLib/raw2mov). It packages the uncompressed video data of specified pixel format into samples of an ISOBMFF file and sets the specific sample entry type as a FOURCC from Table 3. FFmpeg and GStreamer [were used](https://github.com/podborski/isobmff/blob/raw2mov/createRaws.py) to generate the uncompressed video bitstreams which were packaged to ISOBMFF using raw2mov. In addition, Final Cut Pro was used to extract ‘2vuy’ and ‘v210’ QTFF files for verification. Also, the initial version of the [HDRTools](https://gitlab.com/standards/HDRTools/-/tree/feature_mp4) branch was used to verify ‘v210’ code point (note that horiz\_align\_pixels are currently not implemented in HDRTools and the width of the input image has to be a multiple of 48). Then we used 5 different players (QuickTime, MP4Client, ffplay, VLC and playbin) to test the coverage of each code point. Table 3 below summarizes the results.

It can be seen that ‘2vuy’, ‘v210’ and ‘raw ‘ (RGB24) are the most supported code points. Quite the opposite is the support of Digital Voodoo code points from Table 2, only ‘DVOO’ is supported by FFmpeg and all others didn’t play anywhere (not listed in the Table). GStreamer playbin was not able to play any of the files.

Table 3: On overview of player support for playing uncompressed video signaled with the FOURCC in a sample entry.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Code | Description | QT 10.5 (935.5) Mojave | QT 10.5 (1084.4.1) Big Sur | GPAC  1.1.0-DEV-rev114-g923e0e6e9-master | FFMPEG 4.3.1  ffplay | VLC 3.0.11.1 | GStreamer (1.18.0) playbin |
| ['2vuy'](https://mirror.informatimago.com/next/developer.apple.com/quicktime/icefloe/dispatch019.html#2vuy) | 8-bit-per-component 4:2:2 | OK | OK | OK | OK | OK | NO |
| ['yuv2'](https://mirror.informatimago.com/next/developer.apple.com/quicktime/icefloe/dispatch019.html#yuv2) | 8-bit-per-component 4:2:2 | wrong colors after convertion | NO | OK | wrong colors | OK | NO |
| ['v308'](https://mirror.informatimago.com/next/developer.apple.com/quicktime/icefloe/dispatch019.html#v308) | 8-bit-per-component 4:4:4 | NO | NO | wrong mapping | OK | NO | NO |
| ['v408'](https://mirror.informatimago.com/next/developer.apple.com/quicktime/icefloe/dispatch019.html#v408) | 8-bit-per-component 4:4:4:4 | NO | NO | NO | OK | NO | NO |
| ['v216'](https://mirror.informatimago.com/next/developer.apple.com/quicktime/icefloe/dispatch019.html#v216) | 10,12,14,16-bit-per-component 4:2:2 | wrong playback (white), thumbnail is fine | OK | NO | NO | NO | NO |
| ['v410'](https://mirror.informatimago.com/next/developer.apple.com/quicktime/icefloe/dispatch019.html#v410) | 10-bit-per-component 4:4:4 | playback is black, thumbnail wrong colors | playback is black, thumbnail wrong colors | wrong mapping | wrong colors | NO | NO |
| ['v210'](https://mirror.informatimago.com/next/developer.apple.com/quicktime/icefloe/dispatch019.html#v210) | 10-bit-per-component 4:2:2 | OK | OK | wrong mapping | OK | OK | NO |
| 'raw ' | RGB24 8-bit-per-component | OK | OK | OK | OK | OK | NO |
| ‘DVOO’ |  | --- | NO | NO | OK | NO | NO |
| 'j420' | planar YUV 4:2:0, 12bpp, full scale (JPEG) | OK (after convertion) | NO | wrong mapping | NO | NO | NO |
| 'I420' | planar YUV 4:2:0, 12bpp, full scale (JPEG) | NO | NO | NO | OK | OK | NO |
| 'IYUV' | planar YUV 4:2:0, 12bpp | NO | NO | NO | OK | OK | NO |
| 'yv12' | planar YUV 4:2:0, 12bpp | NO | NO | NO | OK | OK | NO |
| 'YVYU' | packed YUV 4:2:2, 16bpp, Y0 Cr Y1 Cb | NO | NO | NO | OK | OK | NO |
| 'RGBA' | packed RGBA 8:8:8:8, 32bpp, RGBARGBA... | OK (after convertion) | NO | NO | OK | OK | NO |
| 'ABGR' | packed ABGR 8:8:8:8, 32bpp, ABGRABGR... | OK (after convertion) | NO | NO | OK | NO | NO |

1. References
2. Jean Le Feuvre, "Raw AV in ISOBMF", Telecom ParisTech, MPEG#122 [m42594](https://dms.mpeg.expert/doc_end_user/documents/122_San%20Diego/wg11/m42594-v1-m42594.zip)
3. Jean Le Feuvre, "Storage of Raw Video in ISOBMFF", Telecom ParisTech, MPEG#132 [m55059](https://dms.mpeg.expert/doc_end_user/documents/132_OnLine/wg11/m55059-v2-m55059.zip)
4. Thomas Stockhammer, "Background and Requirements for Raw Video Format", Qualcomm Incorporated, MPEG#132 [m55286](https://dms.mpeg.expert/doc_end_user/documents/132_OnLine/wg11/m55286-v1-m55286_Raw.zip)
5. Eric Nassor, et al., "Inputs on uncompressed video File Format", Canon Research Centre France, MPEG#133 [m56032](https://dms.mpeg.expert/doc_end_user/documents/133_OnLine/wg11/m56032-v1-m56032_Onuncompressedvideostorage.zip)

1. [Technical Note TN2162](https://developer.apple.com/library/archive/technotes/tn2162/_index.html), "Uncompressed Y´CbCr Video in QuickTime Files", Apple Inc.

1. [QTFF](https://developer.apple.com/library/archive/documentation/QuickTime/QTFF/QTFFPreface/qtffPreface.html), "QuickTime File Format Specification", Apple Inc.
2. "CNN prepping new London facility for launch in March", [link](https://www.newscaststudio.com/2020/02/25/cnn-london-bureau/), Accessed 20 Jan 2021
3. "Inside the RSN’s New ST 2110 IP Production Facility in Wrigleyville", [link](https://www.sportsvideo.org/2020/08/11/marquee-sports-network-spotlight-part-1-inside-the-rsns-new-st-2110-ip-production-facility-in-wrigleyville/), Accessed 20 Jan 2021
4. SMPTE ST 2110-20:2017, “Professional Media Over Managed IP Networks: Uncompressed Active Video”
5. SMPTE ST 377-1:2011, "Material Exchange Format (MXF) — File Format Specification"
6. SMPTE 384M-2005: "Material Exchange Format (MXF) — Mapping of Uncompressed Pictures into the Generic Container"
7. H. Wilkens, G. Hahn, "Simple uncompressed video format for astronomical capturing", version 3, [link](https://free-astro.org/images/5/51/SER_Doc_V3b.pdf), Accessed 22 Jan 2021
8. N. Ouedraogo, J. Le Feuvre, H. Ruellan, F. Denoual, “On uncompressed File Format”, [m57384](https://dms.mpeg.expert/doc_end_user/documents/135_OnLine/wg11/m57384-v1-m57384_UncompressedFileFormat.zip)