

SMPTE REGISTERED DISCLOSURE DOCUMENT

Avid DNxUncompressed – Packing definition and mapping to the MXF Generic Container



Page 1 of 32

The attached document is a Registered Disclosure Document (RDD) prepared by the proponent identified below. It has been examined by the appropriate SMPTE Technology Committee and is believed to contain adequate information to satisfy the objectives defined in the Scope, and to be technically consistent.

This document is NOT a Standard, Recommended Practice or Engineering Guideline, and does NOT imply a finding or representation of the Society.

Every attempt has been made to ensure that the information contained in this document is accurate. Errors in this document should be reported to the proponent identified below, with a copy to eng@smpte.org.

All other inquiries in respect of this document, including inquiries as to intellectual property requirements that may be attached to use of the disclosed technology, should be addressed to the proponent identified below. At this time the submitter is not holding or aware of others holding any patents, and/or applications for patents, that contain Essential Patent Claims, as defined in the SMPTE Intellectual Property Policy, the use of which would be necessary to implement this document

Proponent contact information:

Avid Technology, Inc.
75 Network Drive
Burlington, MA 01803
Attention: DNx Program Office
Email: dnxcodec@avid.com

Table of Contents	Page
1 Scope	5
2 Conformance Notation	5
3 References	5
4 Terms and Definitions	6
4.1 box.....	6
4.2 byte.....	6
4.3 float16.....	6
4.4 float32.....	6
4.5 fourcc.....	6
4.6 Gbps.....	6
4.7 RGB.....	6
4.8 sizeof(<i>op</i>).....	6
4.9 S2.14	6
4.10 uint8	6
4.11 uint10	7
4.12 uint12	7
4.13 uint16	7
4.14 uint16be	7
4.15 uint32be	7
4.16 uint64be	7
4.17 uint16le	7
4.18 uint32le	7
4.19 uint64le	7
4.20 YCbCr	7
4.21 '10.6'	7
4.22 '12.4'	7
5 Packing	8
5.1 'pack'	8
5.2 'fill'	9
5.3 Image essence	9
5.3.1 'sinf'	10
5.3.2 'sdat'	14
5.3.3 'icmp'	14
5.4 Image packing by DNxUncompressed profile types	14
5.4.1 'y208'	14
5.4.2 'y408'	14
5.4.3 'y210'	15
5.4.4 'y410'	15
5.4.5 'y212'	16
5.4.6 'y412'	17
5.4.7 'y216', 'y2S1', 'y2p6', 'y2p4'	18
5.4.8 'y416', 'y4S1', 'y4p6', 'y4p4'	18
5.4.9 '\$20y2h'	18
5.4.10 '\$20y2f'	18
5.4.11 '\$20y4h'	19
5.4.12 '\$20y4f'	19
5.4.13 'rg08'	19
5.4.14 'rg10'	19
5.4.15 'rg12'	20
5.4.16 'rg16', 'rgS1', 'rgp6', 'rgp4'	21

5.4.17	'\$20rgh'	21
5.4.18	'\$20rgf'	21
5.4.19	'\$20a08', '\$20y08', 'cb08', 'cr08', '\$20r08', '\$20g08', '\$20b08'	21
5.4.20	'\$20a16', '\$20y16', 'cb16', 'cr16', '\$20r16', '\$20g16', '\$20b16'	22
5.4.21	'\$20y10', 'cb10', 'cr10', '\$20r10', '\$20g10', '\$20b10'	22
5.4.22	'\$20y12', 'cb12', 'cr12', '\$20r12', '\$20g12', '\$20b12'	23
5.4.23	'\$20\$20yh', '\$20cbh', '\$20crh', '\$20\$20rh', '\$20\$20gh', '\$20\$20bh'	24
5.4.24	'\$20\$20yf', '\$20cbf', '\$20crf', '\$20\$20rf', '\$20\$20gf', '\$20\$20bf'	24
5.4.25	'\$20yS1', '\$20yp6', '\$20yp4', 'cbS1', 'cbp4', 'cbp6', 'crS1', 'crp6', 'crp4', '\$20r16', '\$20rS1', '\$20rp6', '\$20rp4', '\$20g16', '\$20gS1', '\$20gp6', '\$20gp4', '\$20b16', '\$20bS1', '\$20bp6', '\$20bp4'	24
5.4.26	'rl08', 'rl16'	24
6	Essence wrapping of DNxPacked	26
6.1	Frame Wrapping (Informative)	26
6.2	Clip Wrapping (Informative)	26
7	KLV Coding of DNxPacked Coding Units	27
7.1	Essence Element Key	27
7.1.1	Essence Element Count – Byte 14	28
7.1.2	Essence Element Type – Byte 15	28
7.1.3	Essence Element Number – Byte 16	28
7.2	Length	28
7.3	Value	28
7.3.1	Frame-wrapped	28
7.3.2	Clip-wrapped	29
8	SMPTE Label for DNxPacked Essence Container Identification	29
9	SMPTE Label for DNxUncompressed Picture Essence Coding	30
10	Essence Descriptors for DNxUncompressed	31
10.1	Generic Picture Essence Descriptor	31
10.2	CDCI Picture Essence Descriptor	31
10.3	RGBA Picture Essence Descriptor	32
11	Application Issues - Alignment	32
12	Bibliography	32

Introduction

This document defines a generic, structured and extensible wrapping format for data (video, audio and data essence) storage in files, as well as various packing methods for video frames wrapped by this format. This is then mapped into a MXF container. Together these elements form the Avid DNxUncompressed format. The format is targeted at overcoming the limitations and pitfalls of administrating image sequences in production workflows.

The wrapping format is KLV-based and targeted at byte-based processing and storing on standard computer infrastructure. Serial transmission as a bit-stream is not part of the design consideration. The primary goals are the ability to parse the essence structure with minimal coding investment and the ability to introduce new wrapped elements without breaking existing readers.

The packing methods wrapped by the format either use simple binary packing methods or RLE-based compression for lossless storage of the video essence. Simple implementation, speed of access and processing on byte-based computer system is given priority over packing efficiency. All of the methods have been used in this or a similar form before for storage of individual image frames in computer files.

DNXUncompressed is a picture-by-picture coding scheme where each picture is entirely independent and can be extracted as an independent entity. However, the coding units can be simply concatenated to form a sequence of compressed pictures.

The mapping of the image wrapper to MXF overcomes the metadata gap encountered when working with image sequences, where consistency of image coding parameters across the sequence cannot be ensured, frames can be lost, information relevant for video processing but irrelevant for images is either not recorded or needs to be kept in separately with the inherent risk of loss or separation.

This document maps the DNxUncompressed coding units as either frame-wrapped, where each DNxUncompressed frame is individually mapped into a single MXF Generic Container (GC) Picture Item, or clip-wrapped, where a sequence of DNxUncompressed frames is mapped into a single MXF GC Picture Item. It also defines the KLV coding, the Essence Container and Compression Label values and the Essence Descriptor.

1 Scope

This SMPTE Registered Disclosure Document (RDD) specifies a hierarchical packing method for essence data in general (Part 1), as well as a method for the lossless wrapping of image essence using this packing method in MXF files (Part 2). It specifies the mapping of DNxUncompressed Coding Units as a Picture Element that can be used in the Picture Item of the MXF GC. The Picture Element can contain either individual DNxUncompressed frames using frame-wrapping or a sequence of DNxUncompressed frames using clip-wrapping. The MXF Constrained Generic Container (GC) is described in SMPTE ST 379-2.

2 Conformance Notation

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful to the user, but not indispensable, and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except: the Introduction, any section explicitly labeled as "Informative" or individual paragraphs that start with "Note:"

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword "reserved" indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword "forbidden" indicates "reserved" and in addition indicates that the provision will never be defined in the future.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

3 References

SMPTE ST 377-1:2011, Material Exchange Format (MXF) – File Format Specification

SMPTE ST 377-1:2011 Am1:2012, Material Exchange Format (MXF) – File Format Specification – Amendment 1

SMPTE ST 377-1:2011 Am2:2012, Material Exchange Format (MXF) – File Format Specification – Amendment 2

SMPTE ST 379-2:2010, Material Exchange Format (MXF) – MXF Constrained Generic Container

SMPTE ST 400:2012, SMPTE Labels Structure

IEEE Std 754-2008: IEEE standard for floating point arithmetic

4 Terms and Definitions

For the purposes of this document, the following terms and definitions apply.

4.1 box

byte-aligned KLV structure consisting of a fourcc Key, a 32-bit Length and a variable length Value payload

4.2 byte

sequence of 8 bits.

4.3 float16

16-bit (short) floating point, IEEE 754.compliant storage

4.4 float32

32-bit floating point, IEEE 754.compliant storage

4.5 fourcc

32-bit value formed from a sequence of four 8-bit values (aka characters; usually in ASCII coding, but not necessarily), stored in occurrence order: A fourcc 'abcd' would be stored as a sequence `fourcc[4] = {'a', 'b', 'c', 'd'}`, with 'a' residing at the lowest logical memory address. Non-printable characters (e.g. blank ' ' = 20h) are specified by providing their 2 character hexadecimal value, pre-pended by a '\$', e.g. 'xml\$20'.

4.6 Gbps

gigabits per second. The prefix giga (symbol G) is defined in the International System of Units (SI) as a multiplier of 10^9 (1 billion), and therefore 1 gigabit = 10^9 bits = 1000000000 bits.

4.7 RGB

signal storage format using interleaved R, G and B components (see also $YCbCr$). Individual components (non-interleaved) can be stored using planar signal formats. The specifics of the component coding are described in the Picture Essence Descriptors of the MXF container.

4.8 sizeof(*op*)

stored size, in bytes, of the operand *op*.

4.9 S2.14

16-bit signed fixed-point format, allowing 14 bits fractional precision and covering the range $[-2, 2]$. A S2.14 value is stored as a signed 16-bit integer value *v* which is mapped to a float value *f* as $f = ((\text{float})v)/16384$.

4.10 uint8

unsigned 8- bit integer

4.11 uint10

unsigned 10-bit integer

4.12 uint12

unsigned 12-bit integer

4.13 uint16

unsigned 16-bit integer

4.14 uint16be

unsigned 16-bit integer, big-endian

4.15 uint32be

unsigned 32-bit integer, big-endian

4.16 uint64be

unsigned 64-bit integer, big-endian

4.17 uint16le

unsigned 16-bit integer, little-endian

4.18 uint32le

unsigned 32-bit integer, little-endian

4.19 uint64le

unsigned 64-bit integer, little-endian

4.20 YC_BC_R

signal format using an interleaved color difference format (see also RGB). Individual components (non-interleaved) can be stored using planar signal formats. The specifics of the component coding are described in the Picture Essence Descriptors of the MXF container.

4.21 '10.6'

16-bit format storing unsigned 10 bit signal data in the 10 MSBs of the 16-bit value and (optionally) allowing 6 bits for fractional precision. The valid range of values is [0.0, 1023.0], excluding the last 33 values of the 16-bit range (FFC1h-FFFFh). Data are treated and marked as 10 bit.

4.22 '12.4'

16-bit format storing unsigned 12 bit signal data in the 12 MSBs of the 16-bit value and (optionally) allowing 4 bits for fractional precision. The valid range of values is [0.0, 4095.0], excluding the last 15 values of the 16-bit range (FFF1h-FFFFh). Data are treated and marked as 12 bit.

Part 1 – The DNxPacked byte-stream format

5 Packing

The packing format used by DNxUncompressed is a generic byte-stream which consists of KLV box objects. To define these objects this document uses a C++-like pseudo-code notation similar to the ISO base media file format (ISO 14496-12). Elements shall be stored in the order they are defined, byte aligned and close-packed. Derivation implies that all information present in the base class is also present in the derived class, and that, in the bitstream, all such information precedes any additional bitstream syntax declarations specified in the derived class. This generic format is referred to as 'DNxPacked' hereafter.

This document defines the packing types applicable to DNxUncompressed within the DNxPacked structure (see section 5.4).

A box shall consist of a Key-Length `box_header` and a variable-sized payload. The `box_header` shall be defined as

```
box_header(fourcc type)
{
    uint32le    box_size;
    fourcc      box_type = type;
};
```

The `box_size` shall define the outside size of the box, including the header and the payload. The size of a box shall be equal or greater than 8 bytes.

The `box_type` shall define the type of the box and how its payload shall be interpreted.

A *box* object shall be constructed as

```
box(type, payload) : box_header( type )
{
    payload_type box_payload    = payload;
    box_header.box_size        = 8+sizeof(box_payload);
                                // 8: sizeof(box_header)
};
```

The payload shall either be a closely-packed list of 1 to N boxes or directly interpretable fields, but not a mixture of both. A decoder/parser processing a box containing other boxes shall skip any unknown box type and continue processing. To allow future versions of this document to extend the definition of a box without the need to introduce reserved fields, a decoder shall always read the full stored `box_size` and ignore any extra bytes exceeding the size of the payload defined in this document.

5.1 'pack'

```
pack_box: box_header( 'pack' )
{
    box    box_list[];
    box_header.box_size = 8+sum_of(box_list[i].box_size);
};
```

Each essence unit shall be wrapped into a single `pack_box` of type '**pack**', which shall contain a list of other boxes further describing the essence coding. Essence types shall not be mixed within one `pack_box`.

The `pack_box` shall form the coding unit used by the MXF mapping definition.

5.2 'fill'

```
fill_box(fillerSize) : box_header('fill')
{
    uint8 filler[fillerSize];
    boxheader.box_size = 8+fillerSize;
};
```

Fill boxes shall be skipped during parsing. The payload is undefined and shall not be interpreted. Fill boxes can be used, at the encoders discretion, to align following boxes to specific boundaries.

5.3 Image essence

If the `pack_box` contains image essence it shall contain minimum one `signal_data_box` '**sdat**', containing the packed image data, and one `signal_info_box` '**sinf**', describing said image data, as follows:

Each '**sinf**'/'**sdat**' pair shall be packed into a `img_component_box` '**icmp**'. If only a single pair is present the `img_component_box` may be dropped.

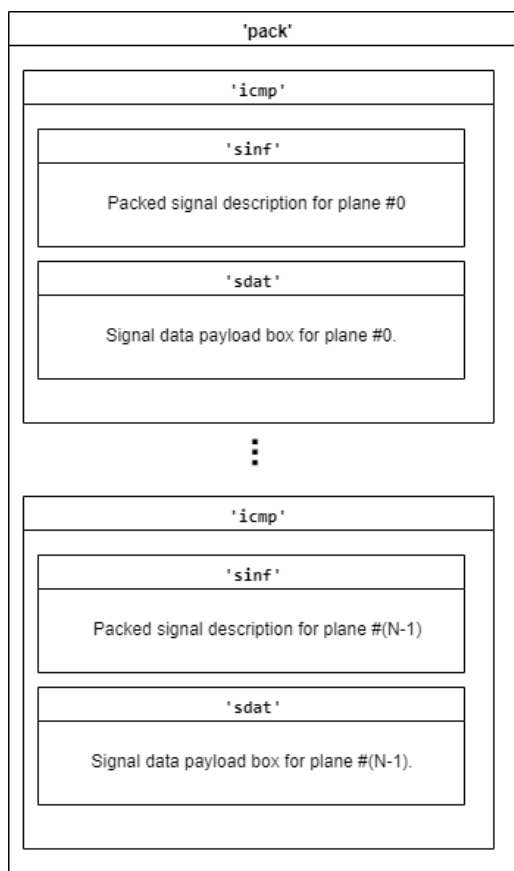


Figure 1 - Packed image essence

5.3.1 'sinf'

```

signal_info_box(width, height, layout, type) : box_header( 'sinf' )
{
    uint32le  frame_width = width;
    uint32le  frame_height= height;
    fourcc     packed_type = type;
    uint8      frame_layout= layout;

    box_header.box_size = 21;
};

```

The `frame_width` and `frame_height` shall describe the sampled region of the coded image. Images shall be stored left-to-right, top-to-bottom. The `packed_type` may define further constraints.

The `frame_layout` shall be 0 for progressive frames (full frame) and 1 for interlaced frames (mixed fields). See SMPTE ST 377-1, Annex G.2.1 for a definition of these layouts.

The `packed_type`, as applicable to DNxUncompressed, is a fourcc profile label from Table 1 or Table 2.