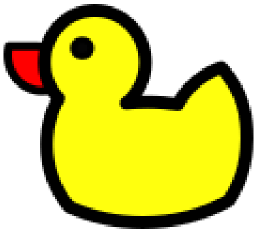
[DUC-FORMAT]:
Desktop Universal Capture Format

Revision History

Revision summary			
Author	**Date**	**Revision history**	**Comments**
Marc-André Moreau	02/24/2017	1.0	Initial draft



Release: February 24th, 2017

Contents

1 Introduction

This document specifies the Desktop Universal Capture (DUC) file format.

1.1 Glossary

The following terms are specific to this document:

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

2 Structures

The following sections specify the DUC file format structures. Unless otherwise specified, all fields defined in this document use the little-endian format. For efficient processing, special care is taken to enforce memory alignment of data structures.

2.1 Common Structures

This section defines common structures.

2.1.1 DUC_PTR

The DUC_PTR structure is used to store a pointer or an offset within a structure.

										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
lo																															
hi																															

lo (4 bytes): An unsigned 32-bit integer containing the low part of a 64-bit unsigned integer.

hi (4 bytes): An unsigned 32-bit integer containing the high part of a 64-bit unsigned integer.

This structure can be interpreted as a single 64-bit unsigned integer. A union type is recommended for accessing either the 32-bit parts or the complete 64-bit value at once.

2.1.2 DUC_POINT

The DUC_POINT structure is used to store the geometric position of a point.

										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
x																y															

x (2 bytes): A signed 16-bit integer containing the x coordinate of the point.

y (2 bytes): A signed 16-bit integer containing the y coordinate of the point.

2.1.3 DUC_SIZE

The DUC_SIZE structure is used to store the size of a rectangle.

										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
width																height															

width (2 bytes): An unsigned 16-bit integer containing the rectangle width.

height (2 bytes): An unsigned 16-bit integer containing the rectangle height.

2.1.4 DUC_RECT

The DUC_POINT structure is used to store the geometric position of a point.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>20</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>30</td><td>1</td></tr>
<tr><td colspan="16">x</td><td colspan="16">y</td></tr>
<tr><td colspan="16">width</td><td colspan="16">height</td></tr>
</table>

x (2 bytes): A signed 16-bit integer containing the x coordinate of the point.

y (2 bytes): A signed 16-bit integer containing the y coordinate of the point.

width (2 bytes): An unsigned 16-bit integer containing the rectangle width.

height (2 bytes): An unsigned 16-bit integer containing the rectangle height.

2.1.5 DUC_PIXEL_CHANNEL

The DUC_PIXEL_CHANNEL structure is used to describe a pixel channel.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>20</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>30</td><td>1</td></tr>
<tr><td colspan="32">name</td></tr>
<tr><td colspan="8">flags</td><td colspan="8">depth</td><td colspan="8">unit</td><td colspan="8">offset</td></tr>
</table>

name (4 bytes): A 4-byte string containing the pixel channel name. The maximum number of characters is 3, and all remaining bytes MUST be set to zero.

flags (1 byte): This field is reserved for future use and MUST be set to zero.

depth (1 byte): The bit depth (number of bits) used to represent the color channel, excluding padding bits.

unit (1 byte): The number of bytes used to store one color channel unit. If the channel is not byte-aligned, then this field MUST be set to zero.

offset (1 byte): The offset of the color channel inside the pixel. If the pixel format is not interleaved or packed, then this field MUST be set to zero.

2.1.6 DUC_PIXEL_FORMAT

The DUC_PIXEL_FORMAT structure is used to encode a pixel format.

<table>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>1
0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>2
0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>3
0</th><th>1</th></tr>
<tr><td colspan="32">name</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="16">id</td><td colspan="16">flags</td></tr>
<tr><td colspan="8">bitsPerPixel</td><td colspan="8">bytesPerPixel</td><td colspan="8">planeFlags</td><td colspan="8">planeCount</td></tr>
<tr><td colspan="8">channelFlags</td><td colspan="8">channelDepth</td><td colspan="8">channelUnit</td><td colspan="8">channelCount</td></tr>
<tr><td colspan="32">subsampling</td></tr>
<tr><td colspan="32">channels</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

name (16 bytes): A 16-byte string containing the pixel format name. The maximum number of characters is 15, and all remaining bytes MUST be set to zero.

id (2 bytes): The pixel format id, with a value in one of the following ranges:

Range	Meaning
0 to 0x3FFF	Static pixel format. This range is reserved for identifiers defined in this specification.
0x3FFF to 0xC000	Dynamic pixel format. This range is used to identify arbitrary pixel formats. Identifiers in this range are subject to change.
0xC000, 0xFFFF	Extended pixel format. For convenience, vendors can define their own pixel format identifiers in this range.

The known (static) pixel formats are the following:

Value	Meaning
DUC_PIXEL_FORMAT_ID_NONE 0x0000	Unknown pixel format.
DUC_PIXEL_FORMAT_ID_ARGB32 0x0001	ARGB32 (A8R8G8B8) pixel format.

Value	Meaning
DUC_PIXEL_FORMAT_ID_XRGB32 0x0002	XRGB32 (X8R8G8B8) pixel format.
DUC_PIXEL_FORMAT_ID_ABGR32 0x0003	ABGR32 (A8B8G8R8) pixel format.
DUC_PIXEL_FORMAT_ID_XBGR32 0x0004	BGRX32 (X8B8G8R8) pixel format.
DUC_PIXEL_FORMAT_ID_BGRA32 0x0005	BGRA32 (B8G8R8A8) pixel format.
DUC_PIXEL_FORMAT_ID_BGRX32 0x0006	BGRX32 (B8G8R8X8) pixel format.
DUC_PIXEL_FORMAT_ID_RGBA32 0x0007	RGBA32 (R8G8B8A8) pixel format.
DUC_PIXEL_FORMAT_ID_RGBX32 0x0008	RGBX32 (R8G8B8X8) pixel format.
DUC_PIXEL_FORMAT_ID_RGB24 0x0009	RGB24 (R8G8B8) pixel format.
DUC_PIXEL_FORMAT_ID_BGR24 0x000A	BGR24 (B8G8R8) pixel format.
DUC_PIXEL_FORMAT_ID_RGB565 0x000B	RGB16 (R5G6B5) pixel format.
DUC_PIXEL_FORMAT_ID_BGR565 0x000C	BGR16 (B5G6R5) pixel format.
DUC_PIXEL_FORMAT_ID_RGB555 0x000D	RGB15 (R5G5B5) pixel format.
DUC_PIXEL_FORMAT_ID_BGR555 0x000E	BGR15 (B5G5R5) pixel format.
DUC_PIXEL_FORMAT_ID_ARGB555 0x000F	ARGB15 (R5G5B5) pixel format.
DUC_PIXEL_FORMAT_ID_BGRA555 0x0010	BGRA15 (B5G5R5) pixel format.
DUC_PIXEL_FORMAT_ID_RGB 0x0020	RGB (planar) pixel format.
DUC_PIXEL_FORMAT_ID_DYNAMIC 0x8000	Dynamic pixel format base id.
DUC_PIXEL_FORMAT_ID_EXTENDED	Extended pixel format base id.

Value	Meaning
0xC000	

The pixel format names use the byte-order naming scheme to avoid any possible confusion that arises with endianness and the word-order naming schemes.

flags (2 bytes): The pixel format flags.

Flag	Meaning
DUC_PIXEL_FORMAT_FLAG_PLANAR 0x0001	Planar pixel order (as opposed to interleaved).
DUC_PIXEL_FORMAT_FLAG_INDEXED 0x0002	Indexed pixel format (palette-based).
DUC_PIXEL_FORMAT_FLAG_PACKED 0x0004	Packed pixel format (like RGB565).
DUC_PIXEL_FORMAT_FLAG_GRAYSCALE 0x0008	Grayscale pixel format (no colors).
DUC_PIXEL_FORMAT_FLAG_RGB 0x0010	RGB pixel format family.
DUC_PIXEL_FORMAT_FLAG_ALPHA 0x0020	The alpha channel is valid.
DUC_PIXEL_FORMAT_FLAG_OPAQUE 0x0040	If the alpha channel is present, it should be made fully opaque (0xFF).
DUC_PIXEL_FORMAT_FLAG_LUMA_CHROMA 0x0080	Luminance (luma) and chrominance (chroma) color space.
DUC_PIXEL_FORMAT_FLAG_SUBSAMPLING 0x0100	Channel subsampling is used.

bitsPerPixel (1 byte): The number of bits per pixel, including any padding bits. This value MUST be byte-aligned (8, 16, 32) if the bytesPerPixel field is not set to zero.

bytesPerPixel (1 byte): The number of bytes per pixel, including any padding bytes. If the pixel is not byte-aligned, then this field MUST be set to zero.

planeFlags (1 byte): This field is reserved for future use and MUST be set to zero.

planeCount (1 byte): The number of pixel planes, usually 3 or 4. This field value MUST be in the [1, 4] range, and MUST NOT exceed the number of color channels.

channelFlags (1 byte): The color channel flags common to all channels.

channelDepth (1 byte): The number of bits per channel, excluding padding bits. If this value is not uniform, then this field MUST be set to zero.

channelUnit (1 byte): The number of bytes used to store one channel unit. If the channel is not byte-aligned, or if this value is not uniform, then this field MUST be set to zero.

channelCount (1 byte): The number of color channels, usually 3 or 4. This field value MUST be in the [1, 4] range.

subsampling (4 bytes): The channel subsampling notation, using one by per color channel. For instance: "4:4:4" (no subsampling) or "4:2:0" (subsampling by half horizontally and vertically). Unused bytes MUST be set to zero.

channels (32 bytes): An array of DUC_PIXEL_CHANNEL structures. The number of elements in this array is specified by the channelCount field. Unused elements MUST be set to zero.

2.1.7 DUC_PIXEL_BUFFER

The DUC_PIXEL_BUFFER structure is used to encode a pixel buffer.

<table>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>20</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>30</th><th>1</th></tr>
<tr><td colspan="32">rect</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">step</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">vstep</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="16">formatId</td><td colspan="16">flags</td></tr>
<tr><td colspan="32">reserved</td></tr>
<tr><td colspan="32">data</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

rect (8 bytes): A DUC_RECT structure containing the position and size of the pixel buffer.

step (8 bytes): An array of four 16-bit unsigned integers containing the step (scanline) for each plane. Unused elements MUST be set to zero.

vstep (8 bytes): An array of four 16-bit unsigned integers containing the vertical step for each plane. Unused elements MUST be set to zero.

formatId (2 bytes): A 16-bit unsigned integer containing the pixel format id.

flags (2 bytes): The pixel buffer flags:

Flag	Meaning
DUC_PIXEL_BUFFER_FLAG_REGION 0x0001	The pixel buffer data represents a region of the full image.
DUC_PIXEL_BUFFER_FLAG_OFFSET 0x0002	The data pointers point to the region of interest (ROI) rather than the image buffer start.
DUC_PIXEL_BUFFER_FLAG_BOTTOM_UP 0x0010	Bottom-up row ordering. The bottom row is first, and the top row is last.

reserved (4 bytes): This field is reserved for future use and MUST be set to zero.

data (32 bytes): An array of 4 DUC_PTR structures pointing to the plane data, in order. The size of each plane in bytes is obtained by multiplying the corresponding step and vstep values.

The following useful variables can be derived from the contents of this structure:

planeSize[i] = **step**[i] x **vstep**[i];

totalSize = **planeSize**[0] + **planeSize**[1] + **planeSize**[2] + **planeSize**[3];

2.2 Block Structures

All block structures begin with a DUC_BLOCK_HEADER structure and finish an alignment pad.

2.2.1 DUC_BLOCK_HEADER

The DUC_BLOCK_HEADER structure is used to encode a block.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="16">blockType</td><td colspan="16">blockFlags</td></tr>
<tr><td colspan="32">blockSize</td></tr>
</table>

blockType (2 bytes): The block type.

Value	Meaning
DUC_HEADER_BLOCK_TYPE 0xDC00	DUC_HEADER_BLOCK
DUC_FORMAT_BLOCK_TYPE 0xDC01	DUC_FORMAT_BLOCK
DUC_SURFACE_BLOCK_TYPE 0xDC02	DUC_SURFACE_BLOCK
DUC_FRAME_BLOCK_TYPE 0xDC03	DUC_FRAME_BLOCK

blockFlags (2 bytes): The block flags, specific to the block type.

Flag	Meaning
DUC_BLOCK_FLAG_COMPRESSED 0x8000	The block is compressed.

blockSize (4 bytes): A 32-bit, unsigned integer containing the total block size, including the size of the header, body and footer. This field value MUST be a multiple of 4.

2.2.2 DUC_ZBLOCK_HEADER

The DUC_ZBLOCK_HEADER structure is used to encode a compressed block.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
blockType																blockFlags															
blockSize																															
compressionType																compressionFlags															
compressedSize																															

blockType (2 bytes): The block type, as defined in DUC_BLOCK_HEADER.

blockFlags (2 bytes): The block flags, as defined in DUC_BLOCK_HEADER.

blockSize (4 bytes): The block size, as defined in DUC_BLOCK_HEADER.

compressionType (2 bytes): The compression type.

compressionFlags (2 bytes): The compression flags.

compressedSize (4 bytes): The compressed data size, excluding headers.

2.2.3 DUC_HEADER_BLOCK

The DUC_HEADER_BLOCK structure is present at the beginning of a DUC file.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
blockType																blockFlags															
blockSize																															
version																padding															
flags																															

reserved1
reserved2
reserved3
reserved4

blockType (2 bytes): The block type, this field MUST be set to DUC_HEADER_BLOCK_TYPE.

blockFlags (2 bytes): This field is reserved for future use and MUST be set to zero.

blockSize (4 bytes): The block size, as defined in DUC_BLOCK_HEADER. This field SHOULD be set to 32.

version (2 bytes): An unsigned 16-bit number containing the format version. The upper 8 bits contain the version major, and the lower 8 bits contain the version minor. This field SHOULD be set to one of the following values:

Value	Meaning
DUC_VERSION_1_0 0x0100	1.0

endianness (2 bytes): this field MUST be set to 0xDC00 (DUC_HEADER_BLOCK_TYPE) using the endianness of the file. Since native endianness is encouraged, this value should be encoded in little-endian in most cases.

flags (4 bytes): This field is reserved for future use and MUST be set to zero.

reserved1 (4 bytes): This field is reserved for future use and MUST be set to zero.

reserved2 (4 bytes): This field is reserved for future use and MUST be set to zero.

reserved3 (4 bytes): This field is reserved for future use and MUST be set to zero.

reserved4 (4 bytes): This field is reserved for future use and MUST be set to zero.

2.2.4 DUC_FORMAT_BLOCK

The DUC_FORMAT_BLOCK structure is used to define pixel formats so that they can be referenced by id in subsequent blocks.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="16">blockType</td><td colspan="16">blockFlags</td></tr>
<tr><td colspan="32">blockSize</td></tr>
<tr><td colspan="32">reserved1</td></tr>
</table>

<table>
<tr><td colspan="16">reserved2</td><td colspan="16">count</td></tr>
<tr><td colspan="32">formats</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

blockType (2 bytes): The block type, this field MUST be set to DUC_FRAME_BLOCK_TYPE.

blockFlags (2 bytes): The block flags, this field is unused and MUST be set to zero.

blockSize (4 bytes): The block size, as defined in DUC_BLOCK_HEADER.

reserved1 (4 bytes): This field is reserved for future use and MUST be set to zero.

reserved2 (2 bytes): This field is reserved for future use and MUST be set to zero.

count (2 bytes): The number of elements in the formats field.

formats (8 bytes): A DUC_PTR structure pointing to an array of DUC_PIXEL_FORMAT structures.

2.2.5 DUC_SURFACE_BLOCK

The DUC_SURFACE_BLOCK structure is used to create, modify or delete a surface.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>20</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>30</td><td>1</td></tr>
<tr><td colspan="16">blockType</td><td colspan="16">blockFlags</td></tr>
<tr><td colspan="32">blockSize</td></tr>
<tr><td colspan="16">surfaceId</td><td colspan="16">formatId</td></tr>
<tr><td colspan="16">surfaceFlags</td><td colspan="16">reserved1</td></tr>
<tr><td colspan="32">surfaceRect</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">reserved2</td></tr>
<tr><td colspan="32">reserved3</td></tr>
</table>

blockType (2 bytes): The block type. This field MUST be set to DUC_SURFACE_BLOCK_TYPE.

blockFlags (2 bytes): This field is unused and MUST be set to zero.

blockSize (4 bytes): The block size, as defined in DUC_BLOCK_HEADER.

surfaceId (2 bytes): A 16-bit unsigned integer containing the corresponding surface id.

formatId (2 bytes): A 16-bit unsigned integer containing the pixel format id.

surfaceFlags (2 bytes): The surface block flags:

Flag	Meaning
DUC_SURFACE_BLOCK_FLAG_CREATE 0x0001	Surface creation.
DUC_SURFACE_BLOCK_FLAG_DELETE 0x0002	Surface deletion.

The DUC_SURFACE_BLOCK_FLAG_CREATE and DUC_SURFACE_BLOCK_FLAG_DELETE flags are mutually exclusive, and the absence of both means a surface modification.

reserved1 (2 bytes): This field is unused and MUST be set to zero.

surfaceRect (8 bytes): A DUC_RECT structure containing the surface position and size.

surfaceTime (4 bytes): A 32-bit unsigned integer containing the timestamp of the surface event.

reserved3 (4 bytes): This field is unused and MUST be set to zero.

2.2.6 DUC_FRAME_BLOCK

The DUC_FRAME_BLOCK structure is used to encode a frame block.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="16">blockType</td><td colspan="16">blockFlags</td></tr>
<tr><td colspan="32">blockSize</td></tr>
<tr><td colspan="16">surfaceId</td><td colspan="16">formatId</td></tr>
<tr><td colspan="32">frameSize</td></tr>
<tr><td colspan="32">frameTime</td></tr>
<tr><td colspan="16">frameId</td><td colspan="8">reserved</td><td colspan="8">count</td></tr>
<tr><td colspan="32">buffers</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

blockType (2 bytes): The block type, this field MUST be set to DUC_FRAME_BLOCK_TYPE.

blockFlags (2 bytes): The block flags, this field is unused and MUST be set to zero.

blockSize (4 bytes): The block size, as defined in DUC_BLOCK_HEADER.

surfaceId (2 bytes): A 16-bit unsigned integer containing the corresponding surface id.

formatId (2 bytes): A 16-bit unsigned integer containing the pixel format id.

frameSize (4 bytes): A DUC_SIZE structure containing the frame size.

frameTime (4 bytes): A 32-bit unsigned integer containing the timestamp of the frame, in milliseconds, relative to the beginning of this capture.

frameId (2 bytes): A 16-bit unsigned integer containing the frame id.

reserved (1 byte): This field is unused and MUST be set to zero.

count (1 byte): The number of elements in the buffers field.

buffers (8 bytes): A DUC_PTR structure pointing to an array of DUC_PIXEL_BUFFER structures.