[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 |



Contents

[1 Introduction 3](#_Toc487717432)

[1.1 Glossary 3](#_Toc487717433)

[2 Structures 4](#_Toc487717434)

[2.1 Common Structures 4](#_Toc487717435)

[2.1.1 DUC\_PTR 4](#_Toc487717436)

[2.1.2 DUC\_POINT 4](#_Toc487717437)

[2.1.3 DUC\_SIZE 4](#_Toc487717438)

[2.1.4 DUC\_RECT 5](#_Toc487717439)

[2.1.5 DUC\_PIXEL\_CHANNEL 5](#_Toc487717440)

[2.1.6 DUC\_PIXEL\_FORMAT 5](#_Toc487717441)

[2.1.7 DUC\_PIXEL\_BUFFER 9](#_Toc487717442)

[2.2 Block Structures 10](#_Toc487717443)

[2.2.1 DUC\_BLOCK\_HEADER 10](#_Toc487717444)

[2.2.2 DUC\_ZBLOCK\_HEADER 11](#_Toc487717445)

[2.2.3 DUC\_HEADER\_BLOCK 11](#_Toc487717446)

[2.2.4 DUC\_FORMAT\_BLOCK 12](#_Toc487717447)

[2.2.5 DUC\_SURFACE\_BLOCK 13](#_Toc487717448)

[2.2.6 DUC\_FRAME\_BLOCK 14](#_Toc487717449)

# Introduction

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

## 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.

# 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.

## Common Structures

This section defines common structures.

### DUC\_PTR

The DUC\_PTR structure is used to store a pointer or an offset within a structure.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 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.

### DUC\_POINT

The DUC\_POINT structure is used to store the geometric position of a point.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 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.

### DUC\_SIZE

The DUC\_SIZE structure is used to store the size of a rectangle.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 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.

### DUC\_RECT

The DUC\_POINT structure is used to store the geometric position of a point.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 0 | 1 |
| x | | | | | | | | | | | | | | | | y | | | | | | | | | | | | | | | |
| width | | | | | | | | | | | | | | | | height | | | | | | | | | | | | | | | |

**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.

### DUC\_PIXEL\_CHANNEL

The DUC\_PIXEL\_CHANNEL structure is used to describe a pixel channel.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 0 | 1 |
| name | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| flags | | | | | | | | depth | | | | | | | | unit | | | | | | | | offset | | | | | | | |

**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.

### DUC\_PIXEL\_FORMAT

The DUC\_PIXEL\_FORMAT structure is used to encode a pixel format.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 0 | 1 |
| name | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| id | | | | | | | | | | | | | | | | flags | | | | | | | | | | | | | | | |
| bitsPerPixel | | | | | | | | bytesPerPixel | | | | | | | | planeFlags | | | | | | | | planeCount | | | | | | | |
| channelFlags | | | | | | | | channelDepth | | | | | | | | channelUnit | | | | | | | | channelCount | | | | | | | |
| subsampling | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| channels | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**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. |
| 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  0xC000 | Extended pixel format base id. |

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](#_DUC_PIXEL_CHANNEL) structures. The number of elements in this array is specified by the channelCount field. Unused elements MUST be set to zero.

### DUC\_PIXEL\_BUFFER

The DUC\_PIXEL\_BUFFER structure is used to encode a pixel buffer.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 0 | 1 |
| rect | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| step | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| vstep | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| formatId | | | | | | | | | | | | | | | | flags | | | | | | | | | | | | | | | |
| reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**rect (8 bytes):** A [DUC\_RECT](#_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](#_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];

## Block Structures

All block structures begin with a [DUC\_BLOCK\_HEADER](#_DUC_BLOCK_HEADER) structure and finish an alignment pad.

### DUC\_BLOCK\_HEADER

The DUC\_BLOCK\_HEADER structure is used to encode a block.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 0 | 1 |
| blockType | | | | | | | | | | | | | | | | blockFlags | | | | | | | | | | | | | | | |
| blockSize | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**blockType (2 bytes):** The block type.

| Value | Meaning |
| --- | --- |
| DUC\_HEADER\_BLOCK\_TYPE  0xDC00 | [DUC\_HEADER\_BLOCK](#_DUC_HEADER_BLOCK) |
| DUC\_FORMAT\_BLOCK\_TYPE  0xDC01 | [DUC\_FORMAT\_BLOCK](#_DUC_FORMAT_BLOCK) |
| DUC\_SURFACE\_BLOCK\_TYPE  0xDC02 | [DUC\_SURFACE\_BLOCK](#_DUC_SURFACE_BLOCK_1) |
| DUC\_FRAME\_BLOCK\_TYPE  0xDC03 | [DUC\_FRAME\_BLOCK](#_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.

### 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 | 1 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 0 | 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.

### 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 | 1 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 0 | 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](#_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.

### 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.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 0 | 1 |
| blockType | | | | | | | | | | | | | | | | blockFlags | | | | | | | | | | | | | | | |
| blockSize | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| reserved1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| reserved2 | | | | | | | | | | | | | | | | count | | | | | | | | | | | | | | | |
| formats | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**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](#_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](#_DUC_PTR) structure pointing to an array of [DUC\_PIXEL\_FORMAT](#_DUC_PIXEL_FORMAT) structures.

### DUC\_SURFACE\_BLOCK

The DUC\_SURFACE\_BLOCK structure is used to create, modify or delete a surface.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 0 | 1 |
| blockType | | | | | | | | | | | | | | | | blockFlags | | | | | | | | | | | | | | | |
| blockSize | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| surfaceId | | | | | | | | | | | | | | | | formatId | | | | | | | | | | | | | | | |
| surfaceFlags | | | | | | | | | | | | | | | | reserved1 | | | | | | | | | | | | | | | |
| surfaceRect | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| reserved2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| reserved3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**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](#_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](#_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.

### DUC\_FRAME\_BLOCK

The DUC\_FRAME\_BLOCK structure is used to encode a frame block.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 0 | 1 |
| blockType | | | | | | | | | | | | | | | | blockFlags | | | | | | | | | | | | | | | |
| blockSize | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| surfaceId | | | | | | | | | | | | | | | | formatId | | | | | | | | | | | | | | | |
| frameSize | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| frameTime | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| frameId | | | | | | | | | | | | | | | | reserved | | | | | | | | count | | | | | | | |
| buffers | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**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](#_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](#_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](#_DUC_PTR) structure pointing to an array of [DUC\_PIXEL\_BUFFER](#_DUC_PIXEL_BUFFER) structures.