

DNG SDK Reference Manual

Generated by Doxygen 1.4.6-NO

Wed Apr 12 13:25:03 2006

Contents

1	Adobe Digital Negative SDK	1
2	DNG SDK Hierarchical Index	2
3	DNG SDK Class Index	4
4	DNG SDK File Index	8
5	DNG SDK Page Index	12
6	DNG SDK Class Documentation	12
7	DNG SDK File Documentation	139
8	DNG SDK Page Documentation	162

1 Adobe Digital Negative SDK

1.1 Introduction

Digital Negative (DNG) is a non-proprietary file format for camera raw image data and metadata. A wide variety of cameras and sensor types are supported by DNG, using the same documented file layout.

This SDK provides support for reading and writing DNG files as well as support for converting DNG data into a displayable or processible image. This SDK is intended to serve as a starting point for adding DNG support to existing applications that use and manipulate images.

1.2 Command line validation: `dng_validate`

A good place to start investigating the DNG SDK is the `dng_validate` command line tool, which can read, validate and convert an existing DNG file. The `dng_validate.cpp` file demonstrates a number of common uses of the SDK. Documentation for the tool can be found [here](#).

1.3 Starting points

- [dng_host](#) Used to customize memory allocation, to communicate progress updates and test for cancellation.
- [dng_negative](#) Main container for metadata and image data in a DNG file.
- [dng_image](#) Class used to hold and manipulate image data.
- [dng_render](#) Class used to convert DNG RAW data to displayable image data.
- [dng_image_writer](#) Class used to write DNG files.

1.4 Related documentation

- The Adobe Digital Negative specification: http://www.adobe.com/products/dng/pdfs/dng_spec.pdf
- TIFF 6 specification: <http://partners.adobe.com/public/developer/tiff/index.html>
- TIFF/EP specification: <http://www.iso.org/iso/en/Catalogue-DetailPage.CatalogueDetail?CSNUMBER=29377&ICS1=37&ICS2=40&ICS3=99>
- EXIF specification: http://www.jeita.or.jp/english/standard/html/1_4.htm
- IPTC specification: <http://www.iptc.org/IPTC7901/>

2 DNG SDK Hierarchical Index

2.1 DNG SDK Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AutoPtr< T >	12
dng_1d_function	15
dng_1d_concatenate	13
dng_1d_identity	18
dng_function_exposure_ramp	50
dng_function_exposure_tone	51
dng_function_gamma_encode	52

dng_function_GammaEncode_1_8	53
dng_function_GammaEncode_2_2	55
dng_function_GammaEncode_sRGB	56
dng_tone_curve_acr3_default	138
dng_1d_table	19
dng_abort_sniffer	20
dng_area_task	22
 dng_filter_task	46
dng_camera_profile	26
dng_color_space	31
 dng_space_AdobeRGB	120
 dng_space_ColorMatch	121
 dng_space_GrayGamma18	122
 dng_space_GrayGamma22	123
 dng_space_ProPhoto	124
 dng_space_sRGB	124
dng_color_spec	33
dng_date_time	36
dng_date_time_storage_info	37
dng_exception	40
dng_exif	41
dng_fingerprint	49
dng_host	58
dng_ifd	63
dng_image	66

dng_simple_image	117
dng_image_writer	69
dng_info	72
dng_iptc	74
dng_linearization_info	76
dng_memory_allocator	79
dng_memory_block	80
dng_memory_data	85
dng_mosaic_info	91
dng_negative	95
dng_pixel_buffer	105
dng_tile_buffer	137
dng_const_tile_buffer	35
dng_dirty_tile_buffer	39
dng_render	113
dng_sniffer_task	118
dng_stream	125
dng_file_stream	44
dng_memory_stream	90

3 DNG SDK Class Index

3.1 DNG SDK Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

[AutoPtr< T >](#) (A class intended to be used in stack scope to hold a pointer from new. The held pointer will be deleted automatically if the scope is left without calling Release on the [AutoPtr](#) first) 12

dng_1d_concatenate (A dng_1d_function that represents the composition (curry) of two other dng_1d_functions)	13
dng_1d_function (A 1D floating-point function)	15
dng_1d_identity (An identity (x -> y such that x == y for all x) mapping function)	18
dng_1d_table (A 1D floating-point lookup table using linear interpolation)	19
dng_abort_sniffer (Class for signaling user cancellation and receiving progress updates)	20
dng_area_task (Abstract class for rectangular processing operations with support for partitioning across multiple processing resources and observing memory constraints)	22
dng_camera_profile (Container for DNG camera color profile and calibration data)	26
dng_color_space (An abstract color space)	31
dng_color_spec (Color transform taking into account white point and camera calibration and individual calibration from DNG negative)	33
dng_const_tile_buffer (Class to get resource acquisition is instantiation behavior for constant (read-only) tile buffers)	35
dng_date_time (Class for holding a date/time and converting to and from relevant date/time formats)	36
dng_date_time_storage_info (Store file offset from which date was read)	37
dng_dirty_tile_buffer (Class to get resource acquisition is instantiation behavior for dirty (writable) tile buffers)	39
dng_exception (All exceptions thrown by the DNG SDK use this exception class)	40
dng_exif (Container class for parsing and holding EXIF tags)	41
dng_file_stream (A stream to/from a disk file. See dng_stream for read/write interface)	44
dng_filter_task (Represents a task which filters an area of a source dng_image to an area of a destination dng_image)	46

dng_fingerprint (Container fingerprint (MD5 only at present))	49
dng_function_exposure_ramp (Curve for pre-exposure-compensation adjustment based on noise floor, shadows, and highlight level)	50
dng_function_exposure_tone (Exposure compensation curve for a given compensation amount in stops using quadric for roll-off)	51
dng_function_gamma_encode (Encoding gamma curve for a given color space)	52
dng_function_GammaEncode_1_8 (A dng_1d_function for gamma encoding with 1.8 gamma)	53
dng_function_GammaEncode_2_2 (A dng_1d_function for gamma encoding with 2.2 gamma)	55
dng_function_GammaEncode_sRGB (A dng_1d_function for gamma encoding in sRGB color space)	56
dng_host (The main class for communication between the application and the DNG SDK. Used to customize memory allocation and other behaviors)	58
dng_ifd (Container for a single image file directory of a digital negative)	63
dng_image (Base class for holding image data in DNG SDK. See dng_simple_image for derived class most often used in DNG SDK)	66
dng_image_writer (Support for writing dng_image or dng_negative instances to a dng_stream in TIFF or DNG format)	69
dng_info (Top-level structure of DNG file with access to metadata)	72
dng_iptc (Class for reading and holding IPTC metadata associated with a DNG file)	74
dng_linearization_info (Class for managing data values related to DNG linearization)	76
dng_memory_allocator (Interface for dng_memory_block allocator)	79
dng_memory_block (Class to provide resource acquisition is instantiation discipline for image buffers and other larger memory allocations)	80
dng_memory_data (Class to provide resource acquisition is instantiation discipline for small memory allocations)	85

dng_memory_stream (A dng_stream which can be read from or written to memory)	90
dng_mosaic_info (Support for describing color filter array patterns and manipulating mosaic sample data)	91
dng_negative (Main class for holding DNG image data and associated metadata)	95
dng_pixel_buffer (Holds a buffer of pixel data with "pixel geometry" meta-data)	105
dng_render (Class used to render digital negative to displayable image)	113
dng_simple_image (Dng_image derived class with simple Trim and Rotate functionality)	117
dng_sniffer_task (Class to establish scope of a named subtask in DNG processing)	118
dng_space_AdobeRGB (Singleton class for AdobeRGB color space)	120
dng_space_ColorMatch (Singleton class for ColorMatch color space)	121
dng_space_GrayGamma18 (Singleton class for gamma 1.8 grayscale color space)	122
dng_space_GrayGamma22 (Singleton class for gamma 2.2 grayscale color space)	123
dng_space_ProPhoto (Singleton class for ProPhoto RGB color space)	124
dng_space_sRGB (Singleton class for sRGB color space)	124
dng_stream (Base stream abstraction. Has support for going between stream and pointer abstraction)	125
dng_tile_buffer (Class to get resource acquisition is instantiation behavior for tile buffers. Can be dirty or constant tile access)	137
dng_tone_curve_acr3_default (Default ACR3 tone curve)	138

4 DNG SDK File Index

4.1 DNG SDK File List

Here is a list of all documented files with brief descriptions:

C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_id_function.h	139
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_id_table.h	139
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_abort_sniffer.h	140
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_area_task.h	140
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_assertions.h	141
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_auto_ptr.h	141
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_bottlenecks.h	142
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_camera_profile.h	147
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_classes.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_space.h	147
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_spec.h	148
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_date_time.h	149
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_errors.h	150
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_exceptions.h	151

C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_exif.h	153
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_fast_ module.h	153
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_file_ stream.h	153
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_ filter_task.h	153
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_ fingerprint.h	154
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_ flags.h	154
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_ globals.h	155
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_ host.h	155
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_ifd.h	155
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_ image.h	156
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_ image_writer.h	156
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_info.h	157
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_iptc.h	158
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_ linearization_info.h	158
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_ lossless_jpeg.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_ matrix.h	158
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_ memory.h	??

C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng - memory_stream.h	159
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng - mosaic_info.h	159
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng - mutex.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng - negative.h	160
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng - orientation.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng - parse_utils.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng - pixel_buffer.h	160
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng - point.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng - rational.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng - read_image.h	160
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_rect.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng - reference.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng - render.h	161
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng - resample.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_sdk - limits.h	161
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng - shared.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng -	

simple_image.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_spline.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_stream.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_string.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_string_list.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_tag_codes.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_tag_types.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_tag_values.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_temperature.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_tile_iterator.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_types.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_utils.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_xmp.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_xmp_sdk.h	??
C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_xy_coord.h	??

5 DNG SDK Page Index

5.1 DNG SDK Related Pages

Here is a list of all related documentation pages:

`doc_dng_validate` ??

6 DNG SDK Class Documentation

6.1 `AutoPtr< T >` Class Template Reference

A class intended to be used in stack scope to hold a pointer from new. The held pointer will be deleted automatically if the scope is left without calling `Release` on the `AutoPtr` first.

```
#include <dng_auto_ptr.h>
```

Public Member Functions

- `AutoPtr ()`
Construct an `AutoPtr` with no referent.
- `AutoPtr (T *p)`
- `~AutoPtr ()`
`Reset()` is called on destruction.
- `void Alloc ()`
Call `Reset` with a pointer from new. Uses `T`'s default constructor.
- `T * Get () const`
Return the owned pointer of this `AutoPtr`, NULL if none. No change in ownership or other effects occur.
- `T * Release ()`
Return the owned pointer of this `AutoPtr`, NULL if none. The `AutoPtr` gives up ownership and takes NULL as its value.
- `void Reset (T *p)`
If a pointer is owned, it is deleted. Ownership is taken of passed in pointer.
- `void Reset ()`

If a pointer is owned, it is deleted and the [AutoPtr](#) takes NULL as its value.

- `T * operator → () const`

Allows members of the owned pointer to be accessed directly. It is an error to call this if the [AutoPtr](#) has NULL as its value.

- `T & operator * () const`

Returns a reference to the object that the owned pointer points to. It is an error to call this if the [AutoPtr](#) has NULL as its value.

6.1.1 Detailed Description

template<class T> class AutoPtr< T >

A class intended to be used in stack scope to hold a pointer from new. The held pointer will be deleted automatically if the scope is left without calling Release on the [AutoPtr](#) first.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 `template<class T> AutoPtr< T >::AutoPtr (T * p) [inline, explicit]`

Construct an [AutoPtr](#) which owns the argument pointer.

Parameters:

- p* pointer which constructed [AutoPtr](#) takes ownership of. *p* will be deleted on destruction or Reset unless Release is called first.

The documentation for this class was generated from the following file:

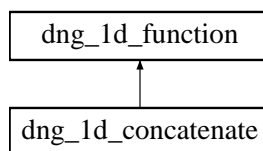
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_auto_ptr.h](#)

6.2 dng_1d_concatenate Class Reference

A [dng_1d_function](#) that represents the composition (curry) of two other dng_1d_ functions.

```
#include <dng_1d_function.h>
```

Inheritance diagram for dng_1d_concatenate::



Public Member Functions

- [dng_1d_concatenate](#) (const [dng_1d_function](#) &function1, const [dng_1d_function](#) &function2)
- virtual bool [IsIdentity](#) () const
Only true if both function1 and function2 have IsIdentity equal to true.
- virtual real64 [Evaluate](#) (real64 x) const
- virtual real64 [EvaluateInverse](#) (real64 y) const

Protected Attributes

- const [dng_1d_function](#) & [fFunction1](#)
- const [dng_1d_function](#) & [fFunction2](#)

6.2.1 Detailed Description

A [dng_1d_function](#) that represents the composition (curry) of two other dng_1d-functions.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 [dng_1d_concatenate::dng_1d_concatenate](#) (const [dng_1d_function](#) & [function1](#), const [dng_1d_function](#) & [function2](#))

Create a [dng_1d_function](#) which computes $y = \text{function2.Evaluate}(\text{function1.Evaluate}(x))$. Compose function1 and function2 to compute $y = \text{function2.Evaluate}(\text{function1.Evaluate}(x))$. The range of function1.Evaluate must be a subset of 0.0 to 1.0 inclusive, otherwise the result of function1(x) will be pinned (clipped) to 0.0 if < 0.0 and to 1.0 if > 1.0 .

Parameters:

- function1* Inner function of composition.
- function2* Outer function of composition.

6.2.3 Member Function Documentation

6.2.3.1 `real64 dng_1d_concatenate::Evaluate (real64 x) const` [virtual]

Return the composed mapping for value x.

Parameters:

x A value between 0.0 and 1.0 (inclusive).

Return values:

function2.Evaluate(function1.Evaluate(x)).

Implements [dng_1d_function](#).

6.2.3.2 `real64 dng_1d_concatenate::EvaluateInverse (real64 y) const` [virtual]

Return the reverse mapped value for y. Be careful using this method with compositions where the inner function does not have a range 0.0 to 1.0 . (Or better yet, do not use such functions.)

Parameters:

y A value to reverse map. Should be within the range of function2.Evaluate.

Return values:

A value x such that `function2.Evaluate(function1.Evaluate(x)) == y` (to very close approximation).

Reimplemented from [dng_1d_function](#).

The documentation for this class was generated from the following files:

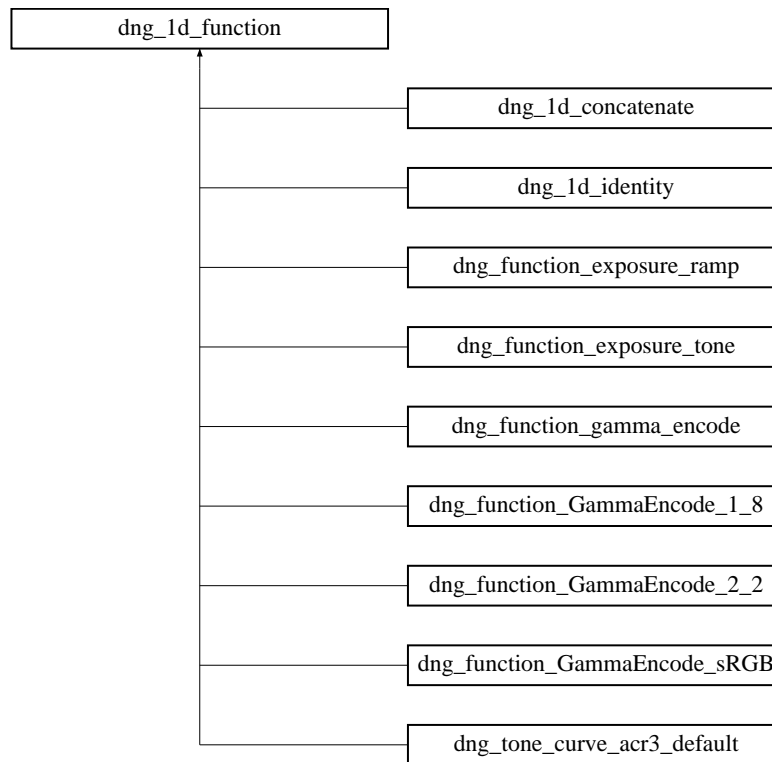
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_1d_function.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_1d_function.cpp

6.3 dng_1d_function Class Reference

A 1D floating-point function.

```
#include <dng_1d_function.h>
```

Inheritance diagram for `dng_1d_function::`



Public Member Functions

- virtual bool [IsIdentity](#) () const
Returns true if this function is the map $x \rightarrow y$ such that $x == y$ for all x . That is if $Evaluate(x) == x$ for all x .
- virtual real64 [Evaluate](#) (real64 x) const =0
- virtual real64 [EvaluateInverse](#) (real64 y) const

6.3.1 Detailed Description

A 1D floating-point function.

The domain (input) is always from 0.0 to 1.0, while the range (output) can be an arbitrary interval.

6.3.2 Member Function Documentation

6.3.2.1 `virtual real64 dng_1d_function::Evaluate (real64 x) const` [pure virtual]

Return the mapping for value x. This method must be implemented by a derived class of [dng_1d_function](#) and the derived class determines the lookup method and function used.

Parameters:

x A value between 0.0 and 1.0 (inclusive).

Return values:

Mapped value for x

Implemented in [dng_1d_identity](#), [dng_1d_concatenate](#), [dng_function_GammaEncode_sRGB](#), [dng_function_GammaEncode_1_8](#), [dng_function_GammaEncode_2_2](#), [dng_function_exposure_ramp](#), [dng_function_exposure_tone](#), [dng_tone_curve_acr3_default](#), and [dng_function_gamma_encode](#).

6.3.2.2 `real64 dng_1d_function::EvaluateInverse (real64 y) const` [virtual]

Return the reverse mapped value for y. This method can be implemented by derived classes. The default implementation uses Newton's method to solve for x such that Evaluate(x) == y.

Parameters:

y A value to reverse map. Should be within the range of the function implemented by this [dng_1d_function](#).

Return values:

A value x such that Evaluate(x) == y (to very close approximation).

Reimplemented in [dng_1d_identity](#), [dng_1d_concatenate](#), [dng_function_GammaEncode_sRGB](#), [dng_function_GammaEncode_1_8](#), [dng_function_GammaEncode_2_2](#), and [dng_tone_curve_acr3_default](#).

The documentation for this class was generated from the following files:

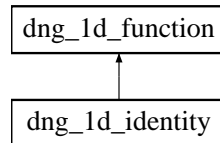
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_1d_function.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_1d_function.cpp

6.4 dng_1d_identity Class Reference

An identity ($x \rightarrow y$ such that $x == y$ for all x) mapping function.

```
#include <dng_1d_function.h>
```

Inheritance diagram for dng_1d_identity::



Public Member Functions

- virtual bool [IsIdentity](#) () const
Always returns true for this class.
- virtual real64 [Evaluate](#) (real64 x) const
Always returns x for this class.
- virtual real64 [EvaluateInverse](#) (real64 y) const
Always returns y for this class.

Static Public Member Functions

- static const [dng_1d_function](#) & [Get](#) ()
This class is a singleton, and is entirely threadsafe. Use this method to get an instance of the class.

6.4.1 Detailed Description

An identity ($x \rightarrow y$ such that $x == y$ for all x) mapping function.

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_1d_function.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_1d_function.cpp

6.5 dng_1d_table Class Reference

A 1D floating-point lookup table using linear interpolation.

```
#include <dng_1d_table.h>
```

Public Types

- enum { **kTableBits** = 12, **kTableSize** = (1 << kTableBits) }
Constants denoting size of table.

Public Member Functions

- void **Initialize** ([dng_memory_allocator](#) &allocator, const [dng_1d_function](#) &function)
- real32 **Interpolate** (real32 x) const

Protected Attributes

- [AutoPtr](#)< [dng_memory_block](#) > **fBuffer**
- real32 * **fTable**

6.5.1 Detailed Description

A 1D floating-point lookup table using linear interpolation.

6.5.2 Member Function Documentation

6.5.2.1 void dng_1d_table::Initialize ([dng_memory_allocator](#) &allocator, const [dng_1d_function](#) &function)

Set up table, initialize entries using function. This method can throw an exception, e.g. if there is not enough memory.

Parameters:

allocator Memory allocator from which table memory is allocated.

function Table is initialized with values of function.Evaluate(0.0) to function.Evaluate(1.0).

6.5.2.2 real32 dng_1d_table::Interpolate (real32 x) const [inline]

Lookup and interpolate mapping for an input.

Parameters:

x value from 0.0 to 1.0 used as input for mapping

Return values:

Approximation of function.Evaluate(*x*)

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_1d_table.h
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_1d_table.cpp

6.6 dng_abort_sniffer Class Reference

Class for signaling user cancellation and receiving progress updates.

```
#include <dng_abort_sniffer.h>
```

Static Public Member Functions

- static void [SniffForAbort](#) ([dng_abort_sniffer](#) *sniffer)

Protected Member Functions

- virtual void [Sniff](#) ()=0
Should be implemented by derived classes to check for an user cancellation.
- virtual void [StartTask](#) (const char *name, real64 fract)
- virtual void [EndTask](#) ()
Signals the end of the innermost task that has been started.
- virtual void [UpdateProgress](#) (real64 fract)

Friends

- class [dng_sniffer_task](#)

6.6.1 Detailed Description

Class for signaling user cancellation and receiving progress updates.

DNG SDK clients should derive a host application specific implementation from this class.

6.6.2 Member Function Documentation

6.6.2.1 `static void dng_abort_sniffer::SniffForAbort (dng_abort_sniffer * sniffer)` [inline, static]

Check for pending user cancellation or other abort. ThrowUserCanceled will be called if one is pending. This static method is provided as a convenience for quickly testing for an abort and throwing an exception if one is pending.

Parameters:

sniffer The dng_sniffer to test for a pending abort. Can be NULL, in which case there an abort is never signalled.

6.6.2.2 `void dng_abort_sniffer::StartTask (const char * name, real64 fract)` [protected, virtual]

Signals the start of a named task with processing in the DNG SDK. Tasks may be nested.

Parameters:

name of the task

fract Percentage of total processing this task is expected to take. From 0.0 to 1.0 .

6.6.2.3 `void dng_abort_sniffer::UpdateProgress (real64 fract)` [protected, virtual]

Signals progress made on current task.

Parameters:

fract percentage of processing completed on current task. From 0.0 to 1.0 .

The documentation for this class was generated from the following files:

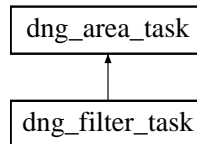
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_abort_sniffer.h
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_abort_sniffer.cpp

6.7 dng_area_task Class Reference

Abstract class for rectangular processing operations with support for partitioning across multiple processing resources and observing memory constraints.

```
#include <dng_area_task.h>
```

Inheritance diagram for dng_area_task::



Public Member Functions

- virtual uint32 [MaxThreads](#) () const
- virtual uint32 [MinTaskArea](#) () const
- virtual dng_point [UnitCell](#) () const
- virtual dng_point [MaxTileSize](#) () const
- virtual dng_rect [RepeatingTile1](#) () const
- virtual dng_rect [RepeatingTile2](#) () const
- virtual dng_rect [RepeatingTile3](#) () const
- virtual void [Start](#) (uint32 threadCount, const dng_point &tileSize, [dng_memory_allocator](#) *allocator, [dng_abort_sniffer](#) *sniffer)
- virtual void [Process](#) (uint32 threadIndex, const dng_rect &tile, [dng_abort_sniffer](#) *sniffer)=0
- virtual void [Finish](#) (uint32 threadCount)
- dng_point [FindTileSize](#) (const dng_rect &area) const
- void [ProcessOnThread](#) (uint32 threadIndex, const dng_rect &area, const dng_point &tileSize, [dng_abort_sniffer](#) *sniffer)

Static Public Member Functions

- static void [Perform](#) ([dng_area_task](#) &task, const dng_rect &area, [dng_memory_allocator](#) *allocator, [dng_abort_sniffer](#) *sniffer)

Protected Attributes

- uint32 **fMaxThreads**
- uint32 **fMinTaskArea**
- dng_point **fUnitCell**
- dng_point **fMaxTileSize**

6.7.1 Detailed Description

Abstract class for rectangular processing operations with support for partitioning across multiple processing resources and observing memory constraints.

6.7.2 Member Function Documentation

6.7.2.1 dng_point dng_area_task::FindTileSize (const dng_rect & *area*) const

Find tile size taking into account repeating tiles, unit cell, and maximum tile size.

Parameters:

area Computation area for which to find tile size.

Return values:

Tile size as height and width in point.

6.7.2.2 void dng_area_task::Finish (uint32 *threadCount*) [virtual]

Task computation finalization and teardown method. Called after all resources have completed processing. Can be overridden to accumulate results and free resources allocated in Start.

Parameters:

threadCount Number of threads used for processing. Same as value passed to Start.

6.7.2.3 virtual uint32 dng_area_task::MaxThreads () const [inline, virtual]

Getter for the maximum number of threads (resources) that can be used for processing

Return values:

Number of threads, minimum of 1, that can be used for this task.

6.7.2.4 virtual dng_point dng_area_task::MaxTileSize () const [inline, virtual]

Getter for maximum size of a tile for processing. Often processing will need to allocate temporary buffers or use other resources that are either fixed or in limited supply. The maximum tile size forces further partitioning if the tile is bigger than this size.

Return values:

Maximum tile size allowed for this area task.

6.7.2.5 `virtual uint32 dng_area_task::MinTaskArea () const` `[inline, virtual]`

Getter for minimum area of a partitioned rectangle. Often it is not profitable to use more resources if it requires partitioning the input into chunks that are too small, as the overhead increases more than the speedup. This method can be overridden for a specific task to indicate the smallest area for partitioning. Default is 256x256 pixels.

Return values:

Minimum area for a partitioned tile in order to give performant operation. (Partitions can be smaller due to small inputs and edge cases.)

6.7.2.6 `void dng_area_task::Perform (dng_area_task & task, const dng_rect & area, dng_memory_allocator * allocator, dng_abort_sniffer * sniffer)` `[static]`

Default resource partitioner that assumes a single resource to be used for processing. Implementations that are aware of multiple processing resources should override (replace) this method. This is usually done in `dng_host::PerformAreaTask`.

Parameters:

task The task to perform.

area The area on which image processing should be performed.

allocator `dng_memory_allocator` to use for allocating temporary buffers, etc.

sniffer `dng_abort_sniffer` to use to check for user cancellation and progress updates.

6.7.2.7 `virtual void dng_area_task::Process (uint32 threadIndex, const dng_rect & tile, dng_abort_sniffer * sniffer)` `[pure virtual]`

Process one tile or fully partitioned area. This method is overridden by derived classes to implement the actual image processing. Note that the sniffer can be ignored if it is certain that a processing task will complete very quickly. This method should never be called directly but rather accessed via `Process`. There is no allocator parameter as all allocation should be done in `Start`.

Parameters:

threadIndex 0 to `threadCount - 1` index indicating which thread this is. (Can be used to get a thread-specific buffer allocated in the `Start` method.)

tile Area to process.

sniffer `dng_abort_sniffer` to use to check for user cancellation and progress updates.

Implemented in `dng_filter_task`.

6.7.2.8 void dng_area_task::ProcessOnThread (uint32 *threadIndex*, const dng_rect & *area*, const dng_point & *tileSize*, [dng_abort_sniffer](#) * *sniffer*)

Handle one resource's worth of partitioned tiles. Called after thread partitioning has already been done. Area may be further subdivided to handle maximum tile size, etc. It will be rare to override this method.

Parameters:

threadIndex 0 to threadCount - 1 index indicating which thread this is.

area Tile area partitioned to this resource.

tileSize

sniffer [dng_abort_sniffer](#) to use to check for user cancellation and progress updates.

6.7.2.9 dng_rect dng_area_task::RepeatingTile1 () const [virtual]

Getter for RepeatingTile1. RepeatingTile1, RepeatingTile2, and RepeatingTile3 are used to establish a set of 0 to 3 tile patterns for which the resulting partitions that the final Process method is called on will not cross tile boundaries in any of the tile patterns. This can be used for a processing routine that needs to read from two tiles and write to a third such that all the tiles are aligned and sized in a certain way. A RepeatingTile value is valid if it is non-empty. Higher numbered RepeatingTile patterns are only used if all lower ones are non-empty. A RepeatingTile pattern must be a multiple of UnitCell in size for all constraints of the partitionerr to be met.

6.7.2.10 dng_rect dng_area_task::RepeatingTile2 () const [virtual]

Getter for RepeatingTile2. RepeatingTile1, RepeatingTile2, and RepeatingTile3 are used to establish a set of 0 to 3 tile patterns for which the resulting partitions that the final Process method is called on will not cross tile boundaries in any of the tile patterns. This can be used for a processing routine that needs to read from two tiles and write to a third such that all the tiles are aligned and sized in a certain way. A RepeatingTile value is valid if it is non-empty. Higher numbered RepeatingTile patterns are only used if all lower ones are non-empty. A RepeatingTile pattern must be a multiple of UnitCell in size for all constraints of the partitionerr to be met.

6.7.2.11 dng_rect dng_area_task::RepeatingTile3 () const [virtual]

Getter for RepeatingTile3. RepeatingTile1, RepeatingTile2, and RepeatingTile3 are used to establish a set of 0 to 3 tile patterns for which the resulting partitions that the final Process method is called on will not cross tile boundaries in any of the tile patterns. This can be used for a processing routine that needs to read from two tiles and write to a third such that all the tiles are aligned and sized in a certain way. A RepeatingTile value is valid if it is non-empty. Higher numbered RepeatingTile patterns are only used

if all lower ones are non-empty. A RepeatingTile pattern must be a multiple of UnitCell in size for all constraints of the partitionerr to be met.

6.7.2.12 void dng_area_task::Start (uint32 threadCount, const dng_point & tileSize, dng_memory_allocator * allocator, dng_abort_sniffer * sniffer)
[virtual]

Task startup method called before any processing is done on partitions. The Start method is called before any processing is done and can be overridden to allocate temporary buffers, etc.

Parameters:

threadCount Total number of threads that will be used for processing. Less than or equal to MaxThreads.

tileSize Size of source tiles which will be processed. (Not all tiles will be this size due to edge conditions.)

allocator dng_memory_allocator to use for allocating temporary buffers, etc.

sniffer Sniffer to test for user cancellation and to set up progress.

Reimplemented in [dng_filter_task](#).

6.7.2.13 virtual dng_point dng_area_task::UnitCell () const [inline, virtual]

Getter for dimensions of which partitioned tiles should be a multiple. Various methods of processing prefer certain alignments. The partitioning attempts to construct tiles such that the sizes are a multiple of the dimensions of this point.

Return values:

a point giving preferred alignment in x and y

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_area_task.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_area_task.cpp

6.8 dng_camera_profile Class Reference

Container for DNG camera color profile and calibration data.

```
#include <dng_camera_profile.h>
```

Public Member Functions

- void [SetName](#) (const char *name)
- const char * [Name](#) () const
- bool [NameIsEmbedded](#) () const
- void [SetCalibrationIlluminant1](#) (uint32 light)
- void [SetCalibrationIlluminant2](#) (uint32 light)
- uint32 [CalibrationIlluminant1](#) () const
- uint32 [CalibrationIlluminant2](#) () const
- real64 [CalibrationTemperature1](#) () const

Getter for first of up to two light sources used for calibration, returning result as color temperature.
- real64 [CalibrationTemperature2](#) () const

Getter for second of up to two light sources used for calibration, returning result as color temperature.
- void [SetColorMatrix1](#) (const dng_matrix &m)
- void [SetColorMatrix2](#) (const dng_matrix &m)
- bool [HasColorMatrix1](#) () const

Predicate to test if first camera matrix is set.
- bool [HasColorMatrix2](#) () const

Predicate to test if second camera matrix is set.
- const dng_matrix & [ColorMatrix1](#) () const

Getter for first of up to two color matrices used for calibrations.
- const dng_matrix & [ColorMatrix2](#) () const

Getter for second of up to two color matrices used for calibrations.
- void [SetCameraCalibration1](#) (const dng_matrix &m)
- void [SetCameraCalibration2](#) (const dng_matrix &m)
- const dng_matrix & [CameraCalibration1](#) () const

Getter for first of up to two color matrices used for individual camera calibrations.
- const dng_matrix & [CameraCalibration2](#) () const

Getter for second of up to two color matrices used for individual camera calibrations.
- void [SetReductionMatrix1](#) (const dng_matrix &m)
- void [SetReductionMatrix2](#) (const dng_matrix &m)
- const dng_matrix & [ReductionMatrix1](#) () const

Getter for first of up to two dimensionality reduction hints for four color cameras.

- const dng_matrix & [ReductionMatrix2](#) () const
Getter for second of up to two dimensionality reduction hints for four color cameras.
- bool [IsValid](#) (uint32 channels) const
- bool [EqualData](#) (const [dng_camera_profile](#) &profile, bool ignore-Calibration=false) const
- void [Parse](#) ([dng_host](#) &host, [dng_stream](#) &stream, [dng_info](#) &info)
Parse profile from DNG file.
- virtual void [SetFourColorBayer](#) ()
Convert from a three-color to a four-color Bayer profile.

Static Protected Member Functions

- static real64 [IlluminantToTemperature](#) (uint32 light)
- static void [NormalizeColorMatrix](#) (dng_matrix &m)

Protected Attributes

- dng_string [fName](#)
- uint32 [fCalibrationIlluminant1](#)
- uint32 [fCalibrationIlluminant2](#)
- dng_matrix [fColorMatrix1](#)
- dng_matrix [fColorMatrix2](#)
- dng_matrix [fCameraCalibration1](#)
- dng_matrix [fCameraCalibration2](#)
- dng_matrix [fReductionMatrix1](#)
- dng_matrix [fReductionMatrix2](#)

6.8.1 Detailed Description

Container for DNG camera color profile and calibration data.

6.8.2 Member Function Documentation

6.8.2.1 uint32 [dng_camera_profile::CalibrationIlluminant1](#) () const
[inline]

Getter for first of up to two light sources used for calibration. Uses the EXIF encodings for illuminant and is used to distinguish which matrix to use. Corresponds to the DNG CalibrationIlluminant1 tag.

6.8.2.2 `uint32 dng_camera_profile::CalibrationIlluminant2 () const` `[inline]`

Getter for second of up to two light sources used for calibration. Uses the EXIF encodings for illuminant and is used to distinguish which matrix to use. Corresponds to the DNG CalibrationIlluminant2 tag.

6.8.2.3 `bool dng_camera_profile::EqualData (const dng_camera_profile & profile, bool ignoreCalibration = false) const`

Predicate to check if two camera profiles are colorwise equal, thus ignores the profile name and optionally the camera calibration.

Parameters:

profile Camera profile to compare to.

ignoreCalibration Controls whether the camera calibration info is checked for equality as well

6.8.2.4 `bool dng_camera_profile::IsValid (uint32 channels) const`

Determines if this a valid profile for this number of color channels?

Return values:

true if the profile is valid.

6.8.2.5 `const char* dng_camera_profile::Name () const` `[inline]`

Getter for camera profile name.

Return values:

Name of profile.

6.8.2.6 `bool dng_camera_profile::NameIsEmbedded () const` `[inline]`

Test if this name is embedded.

Return values:

true if the name matches the name of the embedded camera profile.

6.8.2.7 `void dng_camera_profile::SetCalibrationIlluminant1 (uint32 light)` `[inline]`

Setter for first of up to two light sources used for calibration. Uses the EXIF encodings for illuminant and is used to distinguish which matrix to use. Corresponds to the DNG CalibrationIlluminant1 tag.

6.8.2.8 void dng_camera_profile::SetCalibrationIlluminant2 (uint32 *light*)
[inline]

Setter for second of up to two light sources used for calibration. Uses the EXIF encodings for illuminant and is used to distinguish which matrix to use. Corresponds to the DNG CalibrationIlluminant2 tag.

6.8.2.9 void dng_camera_profile::SetCameraCalibration1 (const dng_matrix & *m*)

Setter for first of up to two color matrices used for individual camera calibrations.

The sequence of matrix transforms is: Camera data → camera calibration → "inverse" of color matrix

This will be a 4x4 matrix for a four-color camera. The defaults are almost always the identity matrix, and for the cases where they aren't, they are diagonal matrices.

6.8.2.10 void dng_camera_profile::SetCameraCalibration2 (const dng_matrix & *m*)

Setter for second of up to two color matrices used for individual camera calibrations.

The sequence of matrix transforms is: Camera data → camera calibration → "inverse" of color matrix

This will be a 4x4 matrix for a four-color camera. The defaults are almost always the identity matrix, and for the cases where they aren't, they are diagonal matrices.

6.8.2.11 void dng_camera_profile::SetColorMatrix1 (const dng_matrix & *m*)

Setter for first of up to two color matrices used for reference camera calibrations. These matrices map XYZ values to camera values. The DNG SDK needs to map colors that direction in order to determine the clipping points for highlight recovery logic based on the white point. If cameras were all three-color, the matrix could be stored as a forward matrix. The inverse matrix is required to support four-color cameras.

6.8.2.12 void dng_camera_profile::SetColorMatrix2 (const dng_matrix & *m*)

Setter for second of up to two color matrices used for reference camera calibrations. These matrices map XYZ values to camera values. The DNG SDK needs to map colors that direction in order to determine the clipping points for highlight recovery logic based on the white point. If cameras were all three-color, the matrix could be stored as a forward matrix. The inverse matrix is required to support four-color cameras.

6.8.2.13 void dng_camera_profile::SetName (const char * *name*) [inline]

Setter for camera profile name.

Parameters:

name Name to use for this camera profile.

6.8.2.14 void dng_camera_profile::SetReductionMatrix1 (const dng_matrix & m)

Setter for first of up to two dimensionality reduction hints for four-color cameras. This is an optional matrix that maps four components to three. See Appendix 6 of the [DNG 1.1.0 specification](#).

6.8.2.15 void dng_camera_profile::SetReductionMatrix2 (const dng_matrix & m)

Setter for second of up to two dimensionality reduction hints for four-color cameras. This is an optional matrix that maps four components to three. See Appendix 6 of the [DNG 1.1.0 specification](#).

The documentation for this class was generated from the following files:

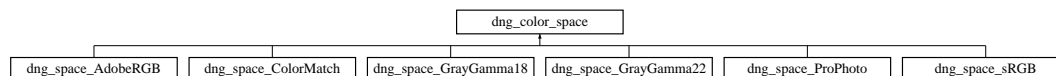
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_camera_profile.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_camera_profile.cpp

6.9 dng_color_space Class Reference

An abstract color space.

```
#include <dng_color_space.h>
```

Inheritance diagram for dng_color_space::



Public Member Functions

- const dng_matrix & [MatrixToPCS](#) () const

Return a matrix which transforms source data in this color space into the Profile Connection Space.

- const dng_matrix & [MatrixFromPCS](#) () const
Return a matrix which transforms Profile Connection Space data into this color space.
- bool [IsMonochrome](#) () const
Predicate which is true if this color space is monochrome (has only a single column).
- virtual const [dng_1d_function](#) & [GammaFunction](#) () const
Getter for the gamma function for this color space.
- bool [IsLinear](#) () const
Returns true if this color space is linear. (I.e. has gamma 1.0.).
- real64 [GammaEncode](#) (real64 x) const
Map an input value through this color space's encoding gamma.
- real64 [GammaDecode](#) (real64 y) const
Map an input value through this color space's decoding gamma (inverse of the encoding gamma).
- virtual bool [ICCProfile](#) (uint32 &size, const uint8 *&data) const

Protected Member Functions

- void [SetMonochrome](#) ()
- void [SetMatrixToPCS](#) (const dng_matrix_3by3 &M)

6.9.1 Detailed Description

An abstract color space.

6.9.2 Member Function Documentation

6.9.2.1 bool [dng_color_space::ICCProfile](#) (uint32 & size, const uint8 *& data)
const [virtual]

Getter for ICC profile, if this color space has one.

Parameters:

- size* Out parameter which receives size on return.
- data* Receives bytes of profile.

Return values:

Returns true if this color space has an ICC profile, false otherwise.

Reimplemented in [dng_space_sRGB](#), [dng_space_AdobeRGB](#), [dng_space_ColorMatch](#), [dng_space_ProPhoto](#), [dng_space_GrayGamma18](#), and [dng_space_GrayGamma22](#).

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_color_space.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_space.cpp

6.10 dng_color_spec Class Reference

Color transform taking into account white point and camera calibration and individual calibration from DNG negative.

```
#include <dng_color_spec.h>
```

Public Member Functions

- [dng_color_spec](#) (const [dng_negative](#) &negative, const [dng_camera_profile](#) *profile)
Read calibration info from DNG negative and construct a [dng_color_spec](#).
- uint32 [Channels](#) () const
Number of channels used for this color transform. Three for most cameras.
- void [SetWhiteXY](#) (const dng_xy_coord &white)
- const dng_xy_coord & [WhiteXY](#) () const
- const dng_vector & [CameraWhite](#) () const
- const dng_matrix & [CameraToPCS](#) () const
- dng_xy_coord [NeutralToXY](#) (const dng_vector &neutral)

6.10.1 Detailed Description

Color transform taking into account white point and camera calibration and individual calibration from DNG negative.

6.10.2 Member Function Documentation

6.10.2.1 const dng_matrix & dng_color_spec::CameraToPCS () const

Getter for camera to Profile Connection Space color transform.

Return values:

A transform that takes into account all camera calibration transforms and white point.

6.10.2.2 const dng_vector & dng_color_spec::CameraWhite () const

Return white point in camera native color coordinates.

Return values:

A dng_vector with components ranging from 0.0 to 1.0 that is normalized such that one component is equal to 1.0 .

6.10.2.3 dng_xy_coord dng_color_spec::NeutralToXY (const dng_vector & *neutral*)

Return the XY value to use for SetWhiteXY for a given camera color space coordinate as the white point.

Parameters:

neutral A camera color space value to use for white point. Components range from 0.0 to 1.0 and should be normalized such that the largest value is 1.0 .

Return values:

White point in XY space that makes neutral map to this XY value as closely as possible.

6.10.2.4 void dng_color_spec::SetWhiteXY (const dng_xy_coord & *white*)

Setter for white point. Value is as XY colorspace coordinate.

Parameters:

white White point to set as an XY value.

6.10.2.5 const dng_xy_coord & dng_color_spec::WhiteXY () const

Getter for white point. Value is as XY colorspace coordinate.

Return values:

XY value of white point.

The documentation for this class was generated from the following files:

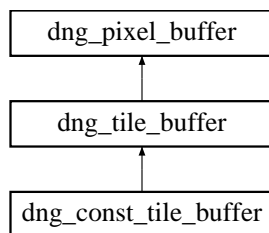
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_spec.h
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_spec.cpp

6.11 dng_const_tile_buffer Class Reference

Class to get resource acquisition is instantiation behavior for constant (read-only) tile buffers.

```
#include <dng_image.h>
```

Inheritance diagram for dng_const_tile_buffer::



Public Member Functions

- [dng_const_tile_buffer](#) (const [dng_image](#) &image, const dng_rect &tile)

6.11.1 Detailed Description

Class to get resource acquisition is instantiation behavior for constant (read-only) tile buffers.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 dng_const_tile_buffer::dng_const_tile_buffer (const [dng_image](#) & *image*, const dng_rect & *tile*)

Obtain a read-only tile from an image.

Parameters:

- image* Image tile will come from.
- tile* Rectangle denoting extent of tile.

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_image.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_image.cpp

6.12 dng_date_time Class Reference

Class for holding a date/time and converting to and from relevant date/time formats.

```
#include <dng_date_time.h>
```

Public Member Functions

- [dng_date_time](#) ()
Construct an invalid date/time.
- [dng_date_time](#) (uint32 year, uint32 month, uint32 day, uint32 hour, uint32 minute, uint32 second)
- bool [IsValid](#) () const
- bool [NotValid](#) () const
- void [Clear](#) ()
Set date to an invalid value.
- bool [Parse](#) (const char *s)

Public Attributes

- uint32 **fYear**
- uint32 **fMonth**
- uint32 **fDay**
- uint32 **fHour**
- uint32 **fMinute**
- uint32 **fSecond**

6.12.1 Detailed Description

Class for holding a date/time and converting to and from relevant date/time formats.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 dng_date_time::dng_date_time (uint32 year, uint32 month, uint32 day, uint32 hour, uint32 minute, uint32 second)

Construct a date/time with specific values.

Parameters:

- year* Year to use as actual integer value, such as 2006.
- month* Month to use from 1 - 12, where 1 is January.

day Day of month to use from 1 -31, where 1 is the first.

hour Hour of day to use from 0 - 23, where 0 is midnight.

minute Minute of hour to use from 0 - 59.

second Second of minute to use from 0 - 59.

6.12.3 Member Function Documentation

6.12.3.1 bool dng_date_time::IsValid () const

Predicate to determine if a date is valid.

Return values:

true if all fields are within range.

6.12.3.2 bool dng_date_time::NotValid () const [inline]

Predicate to determine if a date is invalid.

Return values:

true if any field is out of range.

6.12.3.3 bool dng_date_time::Parse (const char * s)

Parse an EXIF format date string.

Parameters:

s Input date string to parse.

Return values:

true if date was parsed successfully and date is valid.

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_date_time.h
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_date_time.cpp

6.13 dng_date_time_storage_info Class Reference

Store file offset from which date was read.

```
#include <dng_date_time.h>
```

Public Member Functions

- [dng_date_time_storage_info](#) ()
The default constructor initializes to an invalid state.
- [dng_date_time_storage_info](#) (uint32 offset, [dng_date_time_format](#) format)
Construct with file offset and date format.
- bool [IsValid](#) () const
- uint32 [Offset](#) () const
- [dng_date_time_format](#) [Format](#) () const

6.13.1 Detailed Description

Store file offset from which date was read.

Used internally by Adobe to update date in original file.

Warning:

Use at your own risk.

6.13.2 Member Function Documentation

6.13.2.1 [dng_date_time_format](#) [dng_date_time_storage_info::Format](#) () const

Get for format date was originally stored in file. Throws a [dng_error_unknown](#) exception if offset is invalid.

Exceptions:

[dng_exception](#) with `fErrorCode` equal to `dng_error_unknown` if offset is not valid.

6.13.2.2 bool [dng_date_time_storage_info::IsValid](#) () const

Predicate to determine if an offset is valid.

Return values:

true if offset is valid.

6.13.2.3 uint32 [dng_date_time_storage_info::Offset](#) () const

Getter for offset in file.

Exceptions:

[dng_exception](#) with `fErrorCode` equal to `dng_error_unknown` if offset is not valid.

The documentation for this class was generated from the following files:

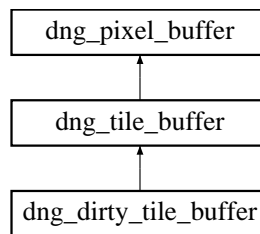
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_date_time.h
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_date_time.cpp

6.14 dng_dirty_tile_buffer Class Reference

Class to get resource acquisition is instantiation behavior for dirty (writable) tile buffers.

```
#include <dng_image.h>
```

Inheritance diagram for dng_dirty_tile_buffer::



Public Member Functions

- [dng_dirty_tile_buffer](#) ([dng_image](#) &image, const dng_rect &tile)

6.14.1 Detailed Description

Class to get resource acquisition is instantiation behavior for dirty (writable) tile buffers.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 dng_dirty_tile_buffer::dng_dirty_tile_buffer ([dng_image](#) & *image*, const dng_rect & *tile*)

Obtain a writable tile from an image.

Parameters:

image Image tile will come from.

tile Rectangle denoting extent of tile.

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_image.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_image.cpp

6.15 dng_exception Class Reference

All exceptions thrown by the DNG SDK use this exception class.

```
#include <dng_exceptions.h>
```

Public Member Functions

- [dng_exception](#) ([dng_error_code](#) code)
- [dng_error_code](#) [ErrorCode](#) () const

6.15.1 Detailed Description

All exceptions thrown by the DNG SDK use this exception class.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 [dng_exception::dng_exception](#) ([dng_error_code](#) code) [inline]

Construct an exception representing the given error code.

Parameters:

code Error code this exception is for.

6.15.3 Member Function Documentation

6.15.3.1 [dng_error_code](#) [dng_exception::ErrorCode](#) () const [inline]

Getter for error code of this exception

Return values:

The error code of this exception.

The documentation for this class was generated from the following file:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_exceptions.h](#)

6.16 dng_exif Class Reference

Container class for parsing and holding EXIF tags.

```
#include <dng_exif.h>
```

Public Member Functions

- void **SetExposureTime** (real64 et, bool snap=true)
- void **SetShutterSpeedValue** (real64 ss)
- void **SetFNumber** (real64 fs)
- void **SetApertureValue** (real64 av)
- void **UpdateDateTime** (const [dng_date_time](#) &dt, int32 tzHour=0x7FFFFFFF)
- virtual bool **ParseTag** ([dng_stream](#) &stream, [dng_shared](#) &shared, uint32 parentCode, bool isMainIFD, uint32 tagCode, uint32 tagType, uint32 tagCount, uint32 tagOffset)
- virtual void **PostParse** ([dng_host](#) &host, [dng_shared](#) &shared)

Static Public Member Functions

- static real64 **SnapExposureTime** (real64 et)
- static [dng_urational](#) **EncodeFNumber** (real64 fs)

Public Attributes

- [dng_string](#) **fImageDescription**
- [dng_string](#) **fMake**
- [dng_string](#) **fModel**
- [dng_string](#) **fSoftware**
- [dng_string](#) **fArtist**
- [dng_string](#) **fCopyright**
- [dng_string](#) **fCopyright2**
- [dng_string](#) **fUserComment**
- [dng_date_time](#) **fDateTime**
- [dng_date_time_storage_info](#) **fDateTimeStorageInfo**
- [dng_date_time](#) **fDateTimeOriginal**
- [dng_date_time](#) **fDateTimeDigitized**
- [dng_string](#) **fSubsecTime**
- [dng_string](#) **fSubsecTimeOriginal**
- [dng_string](#) **fSubsecTimeDigitized**
- int32 **fTimeZoneOffset**
- int32 **fTimeZoneOffsetOriginal**
- uint32 **fTIFF_EP_StandardID**

- uint32 **fExifVersion**
- uint32 **fFlashPixVersion**
- dng_urational **fExposureTime**
- dng_urational **fFNumber**
- dng_srational **fShutterSpeedValue**
- dng_urational **fApertureValue**
- dng_srational **fBrightnessValue**
- dng_srational **fExposureBiasValue**
- dng_urational **fMaxApertureValue**
- dng_urational **fFocalLength**
- dng_urational **fDigitalZoomRatio**
- dng_urational **fExposureIndex**
- dng_urational **fSubjectDistance**
- dng_urational **fGamma**
- dng_urational **fBatteryLevelR**
- dng_string **fBatteryLevelA**
- uint32 **fExposureProgram**
- uint32 **fMeteringMode**
- uint32 **fLightSource**
- uint32 **fFlash**
- uint32 **fFlashMask**
- uint32 **fSensingMethod**
- uint32 **fColorSpace**
- uint32 **fFileSource**
- uint32 **fSceneType**
- uint32 **fCustomRendered**
- uint32 **fExposureMode**
- uint32 **fWhiteBalance**
- uint32 **fSceneCaptureType**
- uint32 **fGainControl**
- uint32 **fContrast**
- uint32 **fSaturation**
- uint32 **fSharpness**
- uint32 **fSubjectDistanceRange**
- uint32 **fSelfTimerMode**
- uint32 **fImageNumber**
- uint32 **fFocalLengthIn35mmFilm**
- uint32 **fISO Speed Ratings** [3]
- uint32 **fSubjectAreaCount**
- uint32 **fSubjectArea** [4]
- uint32 **fComponentsConfiguration**
- dng_urational **fCompressedBitsPerPixel**
- uint32 **fPixelXDimension**

- uint32 **fPixelYDimension**
- dng_urational **fFocalPlaneXResolution**
- dng_urational **fFocalPlaneYResolution**
- uint32 **fFocalPlaneResolutionUnit**
- uint32 **fCFARepeatPatternRows**
- uint32 **fCFARepeatPatternCols**
- uint8 **fCFAPattern** [[kMaxCFAPattern](#)][[kMaxCFAPattern](#)]
- [dng_fingerprint](#) **fImageUniqueID**
- uint32 **fGPSVersionID**
- dng_string **fGPSLatitudeRef**
- dng_urational **fGPSLatitude** [3]
- dng_string **fGPSLongitudeRef**
- dng_urational **fGPSLongitude** [3]
- uint32 **fGPSAltitudeRef**
- dng_urational **fGPSAltitude**
- dng_urational **fGPSTimeStamp** [3]
- dng_string **fGPSSatellites**
- dng_string **fGPSStatus**
- dng_string **fGPSMeasureMode**
- dng_urational **fGPSDOP**
- dng_string **fGPSSpeedRef**
- dng_urational **fGPSSpeed**
- dng_string **fGPSTrackRef**
- dng_urational **fGPSTrack**
- dng_string **fGPSImgDirectionRef**
- dng_urational **fGPSImgDirection**
- dng_string **fGPSMapDatum**
- dng_string **fGPSDestLatitudeRef**
- dng_urational **fGPSDestLatitude** [3]
- dng_string **fGPSDestLongitudeRef**
- dng_urational **fGPSDestLongitude** [3]
- dng_string **fGPSDestBearingRef**
- dng_urational **fGPSDestBearing**
- dng_string **fGPSDestDistanceRef**
- dng_urational **fGPSDestDistance**
- dng_string **fGPSProcessingMethod**
- dng_string **fGPSAreaInformation**
- dng_string **fGPSDateStamp**
- uint32 **fGPSDifferential**
- dng_string **fInteroperabilityIndex**
- uint32 **fInteroperabilityVersion**
- dng_string **fRelatedImageFileFormat**
- uint32 **fRelatedImageWidth**

- uint32 **fRelatedImageLength**
- dng_string **fCameraSerialNumber**
- dng_urational **fLensInfo** [4]
- dng_string **fLensID**
- dng_string **fLensName**
- dng_string **fLensSerialNumber**
- dng_srational **fFlashCompensation**
- dng_string **fOwnerName**
- dng_string **fFirmware**

Protected Member Functions

- virtual bool **Parse_ifd0** ([dng_stream](#) &stream, dng_shared &shared, uint32 parentCode, uint32 tagCode, uint32 tagType, uint32 tagCount, uint32 tagOffset)
- virtual bool **Parse_ifd0_main** ([dng_stream](#) &stream, dng_shared &shared, uint32 parentCode, uint32 tagCode, uint32 tagType, uint32 tagCount, uint32 tagOffset)
- virtual bool **Parse_ifd0_exif** ([dng_stream](#) &stream, dng_shared &shared, uint32 parentCode, uint32 tagCode, uint32 tagType, uint32 tagCount, uint32 tagOffset)
- virtual bool **Parse_gps** ([dng_stream](#) &stream, dng_shared &shared, uint32 parentCode, uint32 tagCode, uint32 tagType, uint32 tagCount, uint32 tagOffset)
- virtual bool **Parse_interoperability** ([dng_stream](#) &stream, dng_shared &shared, uint32 parentCode, uint32 tagCode, uint32 tagType, uint32 tagCount, uint32 tagOffset)

6.16.1 Detailed Description

Container class for parsing and holding EXIF tags.

Public member fields are documented in [EXIF specification](#).

The documentation for this class was generated from the following files:

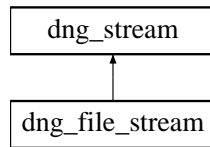
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_exif.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_exif.cpp

6.17 dng_file_stream Class Reference

A stream to/from a disk file. See [dng_stream](#) for read/write interface.

```
#include <dng_file_stream.h>
```

Inheritance diagram for dng_file_stream::



Public Member Functions

- [dng_file_stream](#) (const char *filename, bool output=false)

Protected Member Functions

- virtual uint32 **DoGetLength** ()
- virtual void **DoRead** (void *data, uint32 count, uint32 offset)
- virtual void **DoWrite** (const void *data, uint32 count, uint32 offset)

6.17.1 Detailed Description

A stream to/from a disk file. See [dng_stream](#) for read/write interface.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 dng_file_stream::dng_file_stream (const char * *filename*, bool *output* = false)

Open a stream on a file.

Parameters:

filename Pathname in platform syntax.

output Set to true if writing, false otherwise.

The documentation for this class was generated from the following files:

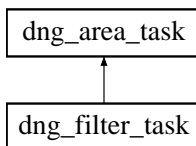
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_file_stream.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_file_stream.cpp

6.18 dng_filter_task Class Reference

Represents a task which filters an area of a source [dng_image](#) to an area of a destination [dng_image](#).

```
#include <dng_filter_task.h>
```

Inheritance diagram for dng_filter_task::



Public Member Functions

- [dng_filter_task](#) (const [dng_image](#) &srcImage, [dng_image](#) &dstImage)
- virtual [dng_rect](#) [SrcArea](#) (const [dng_rect](#) &dstArea)
- virtual [dng_point](#) [SrcTileSize](#) (const [dng_point](#) &dstTileSize)
- virtual void [ProcessArea](#) (uint32 threadIndex, [dng_pixel_buffer](#) &srcBuffer, [dng_pixel_buffer](#) &dstBuffer)=0
- virtual void [Start](#) (uint32 threadCount, const [dng_point](#) &tileSize, [dng_memory_allocator](#) *allocator, [dng_abort_sniffer](#) *sniffer)
- virtual void [Process](#) (uint32 threadIndex, const [dng_rect](#) &area, [dng_abort_sniffer](#) *sniffer)

Protected Attributes

- const [dng_image](#) & [fSrcImage](#)
- [dng_image](#) & [fDstImage](#)
- uint32 [fSrcPlane](#)
- uint32 [fSrcPlanes](#)
- uint32 [fSrcPixelFormat](#)
- uint32 [fDstPlane](#)
- uint32 [fDstPlanes](#)
- uint32 [fDstPixelFormat](#)
- [dng_point](#) [fSrcRepeat](#)
- [AutoPtr](#)< [dng_memory_block](#) > [fSrcBuffer](#) [[kMaxMPThreads](#)]
- [AutoPtr](#)< [dng_memory_block](#) > [fDstBuffer](#) [[kMaxMPThreads](#)]

6.18.1 Detailed Description

Represents a task which filters an area of a source [dng_image](#) to an area of a destination [dng_image](#).

6.18.2 Constructor & Destructor Documentation

6.18.2.1 `dng_filter_task::dng_filter_task (const dng_image & srcImage, dng_image & dstImage)`

Construct a filter task given a source and destination images.

Parameters:

srcImage Image from which source pixels are read.

dstImage Image to which result pixels are written.

6.18.3 Member Function Documentation

6.18.3.1 `void dng_filter_task::Process (uint32 threadIndex, const dng_rect & area, dng_abort_sniffer * sniffer)` [virtual]

Process one tile or partitioned area. Should not be overridden. Instead, override `ProcessArea`, which is where to implement filter processing for a specific type of [dng_filter_task](#). There is no allocator parameter as all allocation should be done in `Start`.

Parameters:

threadIndex 0 to `threadCount` - 1 index indicating which thread this is. (Can be used to get a thread-specific buffer allocated in the `Start` method.)

area Size of tiles to be used for sizing buffers, etc. (Edges of processing can be smaller.)

sniffer [dng_abort_sniffer](#) to use to check for user cancellation and progress updates.

Implements [dng_area_task](#).

6.18.3.2 `virtual void dng_filter_task::ProcessArea (uint32 threadIndex, dng_pixel_buffer & srcBuffer, dng_pixel_buffer & dstBuffer)` [pure virtual]

Implements filtering operation from one buffer to another. Source and destination pixels are set up in member fields of this class. Ideally, no allocation should be done in this routine.

Parameters:

threadIndex The thread on which this routine is being called, between 0 and `threadCount` - 1 for the `threadCount` passed to `Start` method.

srcBuffer Input area and source pixels.

dstBuffer Output area and destination pixels.

6.18.3.3 virtual dng_rect dng_filter_task::SrcArea (const dng_rect & *dstArea*)
[inline, virtual]

Compute the source area needed for a given destination area. Default implementation assumes destination area is equal to source area for all cases.

Parameters:

dstArea Area to for which pixels will be computed.

Return values:

The source area needed as input to calculate the requested destination area.

6.18.3.4 virtual dng_point dng_filter_task::SrcTileSize (const dng_point & *dstTileSize*) [inline, virtual]

Given a destination tile size, calculate input tile size. Similar to SrcArea, and should seldom be overridden.

Parameters:

dstTileSize The destination tile size that is targeted for output.

Return values:

The source tile size needed to compute a tile of the destination size.

6.18.3.5 void dng_filter_task::Start (uint32 *threadCount*, const dng_point & *tileSize*, dng_memory_allocator * *allocator*, dng_abort_sniffer * *sniffer*)
[virtual]

Called prior to processing on specific threads. Can be used to allocate per-thread memory buffers, etc.

Parameters:

threadCount Total number of threads that will be used for processing. Less than or equal to MaxThreads of [dng_area_task](#).

tileSize Size of source tiles which will be processed. (Not all tiles will be this size due to edge conditions.)

allocator [dng_memory_allocator](#) to use for allocating temporary buffers, etc.

sniffer Sniffer to test for user cancellation and to set up progress.

Reimplemented from [dng_area_task](#).

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_filter_task.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_filter_task.cpp

6.19 dng_fingerprint Class Reference

Container fingerprint (MD5 only at present).

```
#include <dng_fingerprint.h>
```

Public Member Functions

- bool [IsNull](#) () const
Check if fingerprint is all zeros.
- bool [IsValid](#) () const
Same as IsNull but expresses intention of testing validity.
- void [Clear](#) ()
Set to all zeros, a value used to indicate an invalid fingerprint.
- bool [operator==](#) (const [dng_fingerprint](#) &print) const
Test if two fingerprints are equal.
- bool [operator!=](#) (const [dng_fingerprint](#) &print) const
Test if two fingerprints are not equal.
- uint32 [Collapse32](#) () const
Produce a 32-bit hash value from fingerprint used for faster hashing of fingerprints.

Public Attributes

- uint8 **data** [16]

6.19.1 Detailed Description

Container fingerprint (MD5 only at present).

The documentation for this class was generated from the following files:

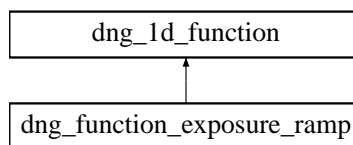
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_fingerprint.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_fingerprint.cpp

6.20 dng_function_exposure_ramp Class Reference

Curve for pre-exposure-compensation adjustment based on noise floor, shadows, and highlight level.

```
#include <dng_render.h>
```

Inheritance diagram for dng_function_exposure_ramp::



Public Member Functions

- **dng_function_exposure_ramp** (real64 white, real64 black, real64 minBlack)
- virtual real64 [Evaluate](#) (real64 x) const

Protected Attributes

- real64 **fSlope**
- real64 **fBlack**
- real64 **fRadius**
- real64 **fQScale**

6.20.1 Detailed Description

Curve for pre-exposure-compensation adjustment based on noise floor, shadows, and highlight level.

6.20.2 Member Function Documentation

6.20.2.1 real64 dng_function_exposure_ramp::Evaluate (real64 x) const [virtual]

Return the mapping for value x . This method must be implemented by a derived class of [dng_1d_function](#) and the derived class determines the lookup method and function used.

Parameters:

x A value between 0.0 and 1.0 (inclusive).

Return values:

Mapped value for x

Implements [dng_1d_function](#).

The documentation for this class was generated from the following files:

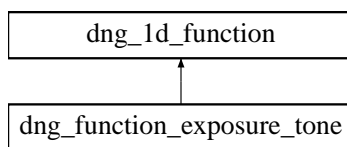
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_render.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_render.cpp

6.21 dng_function_exposure_tone Class Reference

Exposure compensation curve for a given compensation amount in stops using quadric for roll-off.

```
#include <dng_render.h>
```

Inheritance diagram for dng_function_exposure_tone::



Public Member Functions

- **dng_function_exposure_tone** (real64 exposure)
- virtual real64 [Evaluate](#) (real64 x) const

Returns output value for a given input tone.

Protected Attributes

- bool **fIsNOP**
- real64 **fSlope**
- real64 **a**
- real64 **b**
- real64 **c**

6.21.1 Detailed Description

Exposure compensation curve for a given compensation amount in stops using quadric for roll-off.

The documentation for this class was generated from the following files:

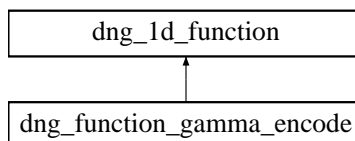
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_render.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_render.cpp

6.22 dng_function_gamma_encode Class Reference

Encoding gamma curve for a given color space.

```
#include <dng_render.h>
```

Inheritance diagram for dng_function_gamma_encode::



Public Member Functions

- **dng_function_gamma_encode** (const [dng_color_space](#) &space)
- virtual real64 **Evaluate** (real64 x) const

Protected Attributes

- const [dng_color_space](#) & **fSpace**

6.22.1 Detailed Description

Encoding gamma curve for a given color space.

6.22.2 Member Function Documentation

6.22.2.1 virtual real64 dng_function_gamma_encode::Evaluate (real64 x) const [virtual]

Return the mapping for value x. This method must be implemented by a derived class of [dng_1d_function](#) and the derived class determines the lookup method and function used.

Parameters:

x A value between 0.0 and 1.0 (inclusive).

Return values:

Mapped value for x

Implements [dng_1d_function](#).

The documentation for this class was generated from the following file:

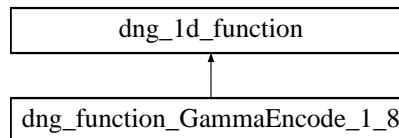
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_render.h](#)

6.23 dng_function_GammaEncode_1_8 Class Reference

A [dng_1d_function](#) for gamma encoding with 1.8 gamma.

```
#include <dng_color_space.h>
```

Inheritance diagram for dng_function_GammaEncode_1_8::



Public Member Functions

- virtual real64 [Evaluate](#) (real64 x) const
- virtual real64 [EvaluateInverse](#) (real64 y) const

Static Public Member Functions

- static const [dng_1d_function](#) & [Get](#) ()

6.23.1 Detailed Description

A [dng_1d_function](#) for gamma encoding with 1.8 gamma.

6.23.2 Member Function Documentation

6.23.2.1 `real64 dng_function_GammaEncode_1_8::Evaluate (real64 x) const` [virtual]

Return the mapping for value x. This method must be implemented by a derived class of [dng_1d_function](#) and the derived class determines the lookup method and function used.

Parameters:

x A value between 0.0 and 1.0 (inclusive).

Return values:

Mapped value for x

Implements [dng_1d_function](#).

6.23.2.2 `real64 dng_function_GammaEncode_1_8::EvaluateInverse (real64 y) const` [virtual]

Return the reverse mapped value for y. This method can be implemented by derived classes. The default implementation uses Newton's method to solve for x such that Evaluate(x) == y.

Parameters:

y A value to reverse map. Should be within the range of the function implemented by this [dng_1d_function](#).

Return values:

A value x such that Evaluate(x) == y (to very close approximation).

Reimplemented from [dng_1d_function](#).

The documentation for this class was generated from the following files:

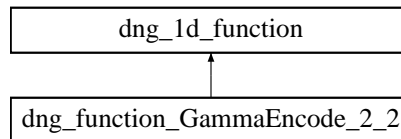
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_color_space.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_space.cpp

6.24 dng_function_GammaEncode_2_2 Class Reference

A [dng_1d_function](#) for gamma encoding with 2.2 gamma.

```
#include <dng_color_space.h>
```

Inheritance diagram for dng_function_GammaEncode_2_2::



Public Member Functions

- virtual real64 [Evaluate](#) (real64 x) const
- virtual real64 [EvaluateInverse](#) (real64 y) const

Static Public Member Functions

- static const [dng_1d_function](#) & [Get](#) ()

6.24.1 Detailed Description

A [dng_1d_function](#) for gamma encoding with 2.2 gamma.

6.24.2 Member Function Documentation

6.24.2.1 real64 dng_function_GammaEncode_2_2::Evaluate (real64 x) const [virtual]

Return the mapping for value x. This method must be implemented by a derived class of [dng_1d_function](#) and the derived class determines the lookup method and function used.

Parameters:

x A value between 0.0 and 1.0 (inclusive).

Return values:

Mapped value for x

Implements [dng_1d_function](#).

6.24.2.2 real64 dng_function_GammaEncode_2_2::EvaluateInverse (real64 y) const [virtual]

Return the reverse mapped value for y. This method can be implemented by derived classes. The default implementation uses Newton's method to solve for x such that Evaluate(x) == y.

Parameters:

- y A value to reverse map. Should be within the range of the function implemented by this [dng_1d_function](#).

Return values:

- A value x such that Evaluate(x) == y (to very close approximation).

Reimplemented from [dng_1d_function](#).

The documentation for this class was generated from the following files:

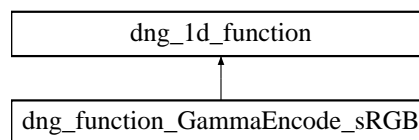
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_color_space.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_space.cpp

6.25 dng_function_GammaEncode_sRGB Class Reference

A [dng_1d_function](#) for gamma encoding in sRGB color space.

```
#include <dng_color_space.h>
```

Inheritance diagram for dng_function_GammaEncode_sRGB::



Public Member Functions

- virtual real64 [Evaluate](#) (real64 x) const
- virtual real64 [EvaluateInverse](#) (real64 y) const

Static Public Member Functions

- static const [dng_1d_function](#) & [Get](#) ()

6.25.1 Detailed Description

A [dng_1d_function](#) for gamma encoding in sRGB color space.

6.25.2 Member Function Documentation

6.25.2.1 `real64 dng_function_GammaEncode_sRGB::Evaluate (real64 x) const` [virtual]

Return the mapping for value x . This method must be implemented by a derived class of [dng_1d_function](#) and the derived class determines the lookup method and function used.

Parameters:

x A value between 0.0 and 1.0 (inclusive).

Return values:

Mapped value for x

Implements [dng_1d_function](#).

6.25.2.2 `real64 dng_function_GammaEncode_sRGB::EvaluateInverse (real64 y) const` [virtual]

Return the reverse mapped value for y . This method can be implemented by derived classes. The default implementation uses Newton's method to solve for x such that $\text{Evaluate}(x) == y$.

Parameters:

y A value to reverse map. Should be within the range of the function implemented by this [dng_1d_function](#).

Return values:

A value x such that $\text{Evaluate}(x) == y$ (to very close approximation).

Reimplemented from [dng_1d_function](#).

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_color_space.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_space.cpp

6.26 dng_host Class Reference

The main class for communication between the application and the DNG SDK. Used to customize memory allocation and other behaviors.

```
#include <dng_host.h>
```

Public Member Functions

- [dng_host](#) ([dng_memory_allocator](#) *allocator=NULL, [dng_abort_sniffer](#) *sniffer=NULL)
- virtual [~dng_host](#) ()

Clean up direct memory for [dng_host](#). Memory allocator and abort sniffer are not deleted. Objects such as [dng_image](#) and others returned from host can still be used after host is deleted.
- [dng_memory_allocator](#) & [Allocator](#) ()

Getter for host's memory allocator.
- virtual [dng_memory_block](#) * [Allocate](#) (uint32 logicalSize)
- [dng_abort_sniffer](#) * [Sniffer](#) ()

Getter for host's abort sniffer.
- virtual void [SniffForAbort](#) ()

Check for pending abort. Should call [ThrowUserCanceled](#) if an abort is pending.
- void [SetNeedsMeta](#) (bool needs)
- bool [NeedsMeta](#) () const

Getter for flag determining whether all XMP metadata should be parsed.
- void [SetNeedsImage](#) (bool needs)
- bool [NeedsImage](#) () const

Setter for flag determining whether DNG image data is needed.
- void [SetForPreview](#) (bool preview)
- bool [ForPreview](#) () const
- void [SetMinimumSize](#) (uint32 size)
- uint32 [MinimumSize](#) () const

Getter for preview minimum pixel width of subsampled image.
- void [SetKeepStage1](#) (bool keep)
- bool [KeepStage1](#) () const

Getter for flag determining whether to keep stage 1, unprocessed raw data, after processing all stages.

- void [SetKeepStage2](#) (bool keep)
- bool [KeepStage2](#) () const
Getter for flag determining whether to keep stage 2, linearized, tone curve processed data, after processing all stages.
- void [SetKeepDNGPrivate](#) (bool keep)
- bool [KeepDNGPrivate](#) ()
Getter for flag determining whether all DNG private data will be kept.
- void [SetKeepOriginalFile](#) (bool keep)
- bool [KeepOriginalFile](#) ()
Getter for flag determining whether to keep original RAW file data.
- virtual bool [IsTransientError](#) (dng_error_code code)
- virtual void [PerformAreaTask](#) (dng_area_task &task, const dng_rect &area)
- virtual dng_exif * [Make_dng_exif](#) ()
Factory method for [dng_exif](#) class. Can be used to customize allocation or to ensure a derived class is used instead of [dng_exif](#).
- virtual dng_shared * [Make_dng_shared](#) ()
Factory method for [dng_shared](#) class. Can be used to customize allocation or to ensure a derived class is used instead of [dng_shared](#).
- virtual dng_ifd * [Make_dng_ifd](#) ()
Factory method for [dng_ifd](#) class. Can be used to customize allocation or to ensure a derived class is used instead of [dng_ifd](#).
- virtual dng_negative * [Make_dng_negative](#) ()
Factory method for [dng_negative](#) class. Can be used to customize allocation or to ensure a derived class is used instead of [dng_negative](#).

6.26.1 Detailed Description

The main class for communication between the application and the DNG SDK. Used to customize memory allocation and other behaviors.

[dng_host](#) allows setting parameters for the DNG conversion, mediates callback style interactions between the host application and the DNG SDK, and allows controlling certain internal behavior of the SDK such as memory allocation. Many applications will be able to use the default implementation of [dng_host](#) by just setting the [dng_memory_allocator](#) and [dng_abort_sniffer](#) in the constructor. More complex interactions will require deriving a class from [dng_host](#).

Multiple [dng_host](#) objects can be allocated in a single process. This may be useful for DNG processing on separate threads. (Distinct [dng_host](#) objects are completely thread-safe for read/write. The application is responsible for establishing mutual exclusion for read/write access to a single [dng_host](#) object if it is used in multiple threads.)

6.26.2 Constructor & Destructor Documentation

6.26.2.1 [dng_host::dng_host](#) ([dng_memory_allocator](#) * *allocator* = NULL, [dng_abort_sniffer](#) * *sniffer* = NULL)

Allocate a [dng_host](#) object, possibly with custom allocator and sniffer.

Parameters:

allocator Allows controlling all memory allocation done via this [dng_host](#). Defaults to singleton global [dng_memory_allocator](#), which calls `new/delete` `dng_malloc_block` for appropriate size.

sniffer Used to periodically check if pending DNG conversions should be aborted and to communicate progress updates. Defaults to singleton global [dng_abort_sniffer](#), which never aborts and ignores progress updated.

6.26.3 Member Function Documentation

6.26.3.1 [dng_memory_block](#) * [dng_host::Allocate](#) (uint32 *logicalSize*) [virtual]

Allocate a new [dng_memory_block](#) using the host's memory allocator. Uses the `Allocator()` property of host to allocate a new block of memory. Will call `ThrowMemoryFull` if block cannot be allocated.

Parameters:

logicalSize Number of usable bytes returned [dng_memory_block](#) must contain.

6.26.3.2 [bool dng_host::ForPreview](#) () const [inline]

Getter for flag determining whether image should be preview quality. Preview quality images may be rendered more quickly. Current DNG SDK does not change rendering behavior based on this flag, but derived versions may use this getter to choose between a slower more accurate path and a faster "good enough for preview" one. Data produce with `ForPreview` set to true should not be written back to a DNG file, except as a preview image.

6.26.3.3 [bool dng_host::IsTransientError](#) ([dng_error_code](#) *code*) [virtual]

Determine if an error is the result of a temporary, but planned-for occurrence such as user cancellation or memory exhaustion. This method is sometimes used to determine whether to try and continue processing a DNG file despite errors in the file format, etc. In such cases, processing will be continued if `IsTransientError` returns false. This is so that user cancellation and memory exhaustion always terminate processing.

Parameters:

code Error to test for transience.

6.26.3.4 void dng_host::PerformAreaTask (dng_area_task & task, const dng_rect & area) [virtual]

General top-level bottleneck for image processing tasks. Default implementation calls `dng_area_task::PerformAreaTask` method on task. Can be overridden in derived classes to support multiprocessing, for example.

Parameters:

task Image processing task to perform on area.

area Rectangle over which to perform image processing task.

6.26.3.5 void dng_host::SetForPreview (bool preview) [inline]

Setter for flag determining whether image should be preview quality, or full quality.

Parameters:

preview If true, rendered images are for preview.

6.26.3.6 void dng_host::SetKeepDNGPrivate (bool keep) [inline]

Setter for flag determining whether DNG private data will be kept.

Parameters:

keep If true, DNG private data will be kept.

6.26.3.7 void dng_host::SetKeepOriginalFile (bool keep) [inline]

Setter for flag determining whether to keep original RAW file data.

Parameters:

keep If true, original RAW data will be kept.

6.26.3.8 void dng_host::SetKeepStage1 (bool *keep*) [inline]

Setter for flag determining whether to keep stage 1, unprocessed raw data, after processing all stages.

Parameters:

keep If true, stage 1 data will be kept.

6.26.3.9 void dng_host::SetKeepStage2 (bool *keep*) [inline]

Setter for flag determining whether to keep stage 2, linearized, tone curve processed data, after processing all stages.

Parameters:

keep If true, stage 2 data will be kept.

6.26.3.10 void dng_host::SetMinimumSize (uint32 *size*) [inline]

Setter for preview minimum pixel width of subsampled image. Used to compute integer binning factor. If set, rendering will use binning to produce an image that is as close to this wide as possible, but no smaller. For example, if the input DNG is 2800 pixels wide and SetMinimumSize is called with 600, a factor of 4 binning will be used and the final image will be 700 pixels wide.

Parameters:

size Target width in pixels for preview image.

6.26.3.11 void dng_host::SetNeedsImage (bool *needs*) [inline]

Setter for flag determining whether DNG image data is needed. Defaults to true. Image data might not be needed for applications which only manipulate metadata.

Parameters:

needs If true, image data is needed.

6.26.3.12 void dng_host::SetNeedsMeta (bool *needs*) [inline]

Setter for flag determining whether all XMP metadata should be parsed. Defaults to true. One might not want metadata when doing a quick check to see if a file is readable.

Parameters:

needs If true, metadata is needed.

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_host.h
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_host.cpp

6.27 dng_ifd Class Reference

Container for a single image file directory of a digital negative.

```
#include <dng_ifd.h>
```

Public Types

- enum { **kMaxTileInfo** = 32 }

Public Member Functions

- virtual bool **ParseTag** (dng_stream &stream, uint32 parentCode, uint32 tagCode, uint32 tagType, uint32 tagCount, uint32 tagOffset)
- virtual void **PostParse** ()
- virtual bool **IsValidDNG** (dng_shared &shared, uint32 parentCode)
- dng_rect **Bounds** () const
- uint32 **TilesAcross** () const
- uint32 **TilesDown** () const
- uint32 **TilesPerImage** () const
- dng_rect **TileArea** (uint32 rowIndex, uint32 colIndex) const
- virtual uint32 **TileByteCount** (const dng_rect &tile) const
- void **SetSingleStrip** ()
- void **FindTileSize** (uint32 bytesPerTile=128 *1024, uint32 cellH=16, uint32 cellV=16)
- void **FindStripSize** (uint32 bytesPerStrip=128 *1024, uint32 cellV=16)
- virtual uint32 **PixelType** () const
- virtual bool **IsBaselineJPEG** () const
- virtual bool **CanRead** () const
- virtual void **ReadImage** (dng_host &host, dng_stream &stream, dng_image &image) const

Public Attributes

- bool **fUsesNewSubFileType**
- uint32 **fNewSubFileType**
- uint32 **fImageWidth**

- uint32 **fImageLength**
- uint32 **fBitsPerSample** [[kMaxSamplesPerPixel](#)]
- uint32 **fCompression**
- uint32 **fPredictor**
- uint32 **fPhotometricInterpretation**
- uint32 **fFillOrder**
- uint32 **fOrientation**
- uint32 **fSamplesPerPixel**
- uint32 **fPlanarConfiguration**
- real64 **fXResolution**
- real64 **fYResolution**
- uint32 **fResolutionUnit**
- bool **fUsesStrips**
- bool **fUsesTiles**
- uint32 **fTileWidth**
- uint32 **fTileLength**
- uint32 **fTileOffsetsType**
- uint32 **fTileOffsetsCount**
- uint32 **fTileOffsetsOffset**
- uint32 **fTileOffset** [[kMaxTileInfo](#)]
- uint32 **fTileByteCountsType**
- uint32 **fTileByteCountsCount**
- uint32 **fTileByteCountsOffset**
- uint32 **fTileByteCount** [[kMaxTileInfo](#)]
- uint32 **fSubIFDsCount**
- uint32 **fSubIFDsOffset**
- uint32 **fExtraSamplesCount**
- uint32 **fExtraSamples** [[kMaxSamplesPerPixel](#)]
- uint32 **fSampleFormat** [[kMaxSamplesPerPixel](#)]
- uint32 **fJPEGTablesCount**
- uint32 **fJPEGTablesOffset**
- uint32 **fJPEGInterchangeFormat**
- uint32 **fJPEGInterchangeFormatLength**
- real64 **fYCbCrCoefficientR**
- real64 **fYCbCrCoefficientG**
- real64 **fYCbCrCoefficientB**
- uint32 **fYCbCrSubSampleH**
- uint32 **fYCbCrSubSampleV**
- uint32 **fYCbCrPositioning**
- real64 **fReferenceBlackWhite** [6]
- uint32 **fCFARepeatPatternRows**
- uint32 **fCFARepeatPatternCols**
- uint8 **fCFAPattern** [[kMaxCFAPattern](#)][[kMaxCFAPattern](#)]

- uint8 **fCFAPlaneColor** [[kMaxColorPlanes](#)]
- uint32 **fCFALayout**
- uint32 **fLinearizationTableType**
- uint32 **fLinearizationTableCount**
- uint32 **fLinearizationTableOffset**
- uint32 **fBlackLevelRepeatRows**
- uint32 **fBlackLevelRepeatCols**
- real64 **fBlackLevel** [[kMaxBlackPattern](#)][[kMaxBlackPattern](#)][[kMaxSamplesPerPixel](#)]
- uint32 **fBlackLevelDeltaHType**
- uint32 **fBlackLevelDeltaHCount**
- uint32 **fBlackLevelDeltaHOffset**
- uint32 **fBlackLevelDeltaVType**
- uint32 **fBlackLevelDeltaVCount**
- uint32 **fBlackLevelDeltaVOffset**
- real64 **fWhiteLevel** [[kMaxSamplesPerPixel](#)]
- dng_urational **fDefaultScaleH**
- dng_urational **fDefaultScaleV**
- dng_urational **fBestQualityScale**
- dng_urational **fDefaultCropOriginH**
- dng_urational **fDefaultCropOriginV**
- dng_urational **fDefaultCropSizeH**
- dng_urational **fDefaultCropSizeV**
- uint32 **fBayerGreenSplit**
- dng_urational **fChromaBlurRadius**
- dng_urational **fAntiAliasStrength**
- dng_rect **fActiveArea**
- uint32 **fMaskedAreaCount**
- dng_rect **fMaskedArea** [[kMaxMaskedAreas](#)]
- bool **fLosslessJPEGBug16**
- uint32 **fSampleBitShift**
- uint32 **fThisIFD**
- uint32 **fNextIFD**

Protected Member Functions

- virtual bool **IsValidCFA** (dng_shared &shared, uint32 parentCode)

6.27.1 Detailed Description

Container for a single image file directory of a digital negative.

See [DNG 1.1.0 specification](#) for documentation of specific tags.

The documentation for this class was generated from the following files:

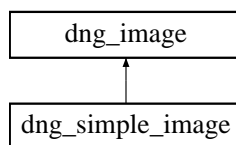
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_ifd.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_ifd.cpp

6.28 dng_image Class Reference

Base class for holding image data in DNG SDK. See [dng_simple_image](#) for derived class most often used in DNG SDK.

```
#include <dng_image.h>
```

Inheritance diagram for dng_image::



Public Types

- enum [edge_option](#) { [edge_none](#), [edge_zero](#), [edge_repeat](#), [edge_repeat_zero_last](#) }

How to handle requests to get image areas outside the image bounds.

Public Member Functions

- const dng_rect & [Bounds](#) () const
Getter method for bounds of an image.
- dng_point [Size](#) () const
Getter method for size of an image.
- uint32 [Width](#) () const
Getter method for width of an image.

- uint32 [Height](#) () const
Getter method for height of an image.
- uint32 [Planes](#) () const
Getter method for number of planes in an image.
- uint32 [PixelType](#) () const
- uint32 [PixelSize](#) () const
- uint32 [PixelRange](#) () const
- virtual dng_rect [RepeatingTile](#) () const
Getter for best "tile stride" for accessing image.
- void [Get](#) (dng_pixel_buffer &buffer, [edge_option](#) edgeOption=edge_none, uint32 repeatV=1, uint32 repeatH=1) const
- void [Put](#) (const dng_pixel_buffer &buffer)
- virtual void [Trim](#) (const dng_rect &r)
- virtual void [Rotate](#) (const dng_orientation &orientation)

Protected Member Functions

- **dng_image** (const dng_rect &bounds, uint32 planes, uint32 pixelType, uint32 pixelRange)
- virtual void **AcquireTileBuffer** (dng_pixel_buffer &buffer, const dng_rect &tile, bool dirty) const
- virtual void **ReleaseTileBuffer** (dng_pixel_buffer &buffer) const
- virtual void **DoGet** (dng_pixel_buffer &buffer) const
- virtual void **DoPut** (const dng_pixel_buffer &buffer)
- void **GetRepeat** (dng_pixel_buffer &buffer, const dng_rect &srcArea, const dng_rect &dstArea) const
- void **GetEdge** (dng_pixel_buffer &buffer, [edge_option](#) edgeOption, const dng_rect &srcArea, const dng_rect &dstArea) const

Protected Attributes

- dng_rect **fBounds**
- uint32 **fPlanes**
- uint32 **fPixelType**
- uint32 **fPixelRange**

Friends

- class [dng_tile_buffer](#)

6.28.1 Detailed Description

Base class for holding image data in DNG SDK. See [dng_simple_image](#) for derived class most often used in DNG SDK.

6.28.2 Member Enumeration Documentation

6.28.2.1 enum [dng_image::edge_option](#)

How to handle requests to get image areas outside the image bounds.

Enumerator:

- edge_none* Leave edge pixels unchanged.
- edge_zero* Pad with zeros.
- edge_repeat* Repeat edge pixels.
- edge_repeat_zero_last* Repeat edge pixels, except for last plane which is zero padded.

6.28.3 Member Function Documentation

6.28.3.1 void [dng_image::Get](#) ([dng_pixel_buffer](#) & *buffer*, [edge_option](#) *edgeOption* = *edge_none*, [uint32](#) *repeatV* = 1, [uint32](#) *repeatH* = 1) const

Get a pixel buffer of data on image with proper edge padding.

Parameters:

- buffer* Receives resulting pixel buffer.
- edgeOption* [edge_option](#) describing how to pad edges.
- repeatV* Amount of repeated padding needed in vertical for *edge_repeat* and *edge_repeat_zero_last* *edgeOption* cases.
- repeatH* Amount of repeated padding needed in horizontal for *edge_repeat* and *edge_repeat_zero_last* *edgeOption* cases.

6.28.3.2 [uint32 dng_image::PixelRange \(\)](#) const

Getter for pixel range. For unsigned types, range is 0 to return value. For signed types, range is return value - 0x8000U. For tFloat type, pixel range is 0.0 to 1.0 and this routine returns 1.

6.28.3.3 [uint32 dng_image::PixelSize \(\)](#) const

Getter for pixel size.

Return values:

Size, in bytes, of pixel type for this image .

6.28.3.4 uint32 dng_image::PixelType () const [inline]

Getter for pixel type.

Return values:

See dng_tagtypes.h . Valid values are ttByte, ttShort, ttSShort, ttLong, ttFloat .

6.28.3.5 void dng_image::Put (const dng_pixel_buffer & buffer)

Put a pixel buffer into image.

Parameters:

buffer Pixel buffer to copy from.

6.28.3.6 void dng_image::Rotate (const dng_orientation & orientation) [virtual]

Rotate image to reflect given orientation change.

Parameters:

orientation Directive to rotate image in a certain way.

Reimplemented in [dng_simple_image](#).

6.28.3.7 void dng_image::Trim (const dng_rect & r) [virtual]

Shrink bounds of image to given rectangle.

Parameters:

r Rectangle to crop to.

Reimplemented in [dng_simple_image](#).

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_image.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_image.cpp

6.29 dng_image_writer Class Reference

Support for writing [dng_image](#) or [dng_negative](#) instances to a [dng_stream](#) in TIFF or DNG format.

```
#include <dng_image_writer.h>
```

Public Member Functions

- virtual void **WriteTIFF** (dng_host &host, dng_stream &stream, const dng_image &image, uint32 photometricInterpretation=piBlackIsZero, uint32 compression=ccUncompressed, const dng_negative *negative=NULL, const dng_color_space *space=NULL, const dng_resolution *resolution=NULL, const dng_jpeg_preview *thumbnail=NULL, const dng_memory_block *imageResources=NULL)
- virtual void **WriteDNG** (dng_host &host, dng_stream &stream, const dng_negative &negative, const dng_image &thumbnail, uint32 compression=cc-JPEG, const dng_jpeg_preview *preview1=NULL, const dng_jpeg_preview *preview2=NULL)

Protected Types

- enum { **kImageBufferSize** = 128 * 1024 }

Protected Member Functions

- virtual uint32 **CompressedBufferSize** (const dng_ifd &ifd, uint32 uncompressedSize)
- virtual void **EncodePredictor** (dng_host &host, const dng_ifd &ifd, dng_pixel_buffer &buffer)
- virtual void **ByteSwapBuffer** (dng_host &host, dng_pixel_buffer &buffer)
- virtual void **WriteData** (dng_host &host, const dng_ifd &ifd, dng_stream &stream, dng_pixel_buffer &buffer)
- virtual void **WriteTile** (dng_host &host, const dng_ifd &ifd, dng_stream &stream, const dng_image &image, const dng_rect &tileArea, bool mapRange, uint32 fakeChannels)
- virtual void **WriteImage** (dng_host &host, const dng_ifd &ifd, basic_tag_set &basic, dng_stream &stream, const dng_image &image, bool mapRange=false, uint32 fakeChannels=1)

Protected Attributes

- [AutoPtr< dng_memory_block >](#) **fCompressedBuffer**
- [AutoPtr< dng_memory_block >](#) **fUncompressedBuffer**

6.29.1 Detailed Description

Support for writing [dng_image](#) or [dng_negative](#) instances to a [dng_stream](#) in TIFF or DNG format.

6.29.2 Member Function Documentation

6.29.2.1 void `dng_image_writer::WriteDNG` (`dng_host` & *host*, `dng_stream` & *stream*, const `dng_negative` & *negative*, const `dng_image` & *thumbnail*, uint32 *compression* = `ccJPEG`, const `dng_jpeg_preview` * *preview1* = NULL, const `dng_jpeg_preview` * *preview2* = NULL) [virtual]

Write a `dng_image` to a `dng_stream` in DNG format.

Parameters:

- host* Host interface used for progress updates, abort testing, buffer allocation, etc.
- stream* The `dng_stream` on which to write the TIFF.
- negative* The image data and metadata (EXIF, IPTC, XMP) to be written.
- thumbnail* Thumbanil image. Must be provided.
- compression* Either `ccUncompressed` or `ccJPEG` for lossless JPEG.
- preview1* First preview image to write.
- preview2* Second preview image to write. Ignored if *preview1* is NULL.

6.29.2.2 void `dng_image_writer::WriteTIFF` (`dng_host` & *host*, `dng_stream` & *stream*, const `dng_image` & *image*, uint32 *photometricInterpretation* = `piBlackIsZero`, uint32 *compression* = `ccUncompressed`, const `dng_negative` * *negative* = NULL, const `dng_color_space` * *space* = NULL, const `dng_resolution` * *resolution* = NULL, const `dng_jpeg_preview` * *thumbnail* = NULL, const `dng_memory_block` * *imageResources* = NULL) [virtual]

Write a `dng_image` to a `dng_stream` in TIFF format.

Parameters:

- host* Host interface used for progress updates, abort testing, buffer allocation, etc.
- stream* The `dng_stream` on which to write the TIFF.
- image* The actual image data to be written.
- photometricInterpretation* Either `piBlackIsZero` for monochrome or `piRGB` for RGB images.
- compression* Must be `ccUncompressed` .
- negative* If non-NULL, EXIF, IPTC, and XMP metadata from this negative is written to TIFF.
- space* If non-null and color space has an ICC profile, TIFF will be tagged with this profile. No color space conversion of image data occurs.
- resolution* If non-NULL, TIFF will be tagged with this resolution.
- thumbnail* If non-NULL, will be stored in TIFF as preview image.
- imageResources* If non-NULL, will image resources be stored in TIFF as well.

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_image_writer.h
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_image_writer.cpp

6.30 dng_info Class Reference

Top-level structure of DNG file with access to metadata.

```
#include <dng_info.h>
```

Public Member Functions

- virtual void [Parse](#) ([dng_host](#) &host, [dng_stream](#) &stream)
- virtual void [PostParse](#) ([dng_host](#) &host)
Must be called immediately after a successful Parse operation.
- virtual bool [IsValidDNG](#) ()

Public Attributes

- uint32 [fTIFFBlockOffset](#)
- bool [fBigEndian](#)
- uint32 [fMagic](#)
- [AutoPtr](#)< [dng_exif](#) > [fExif](#)
- [AutoPtr](#)< [dng_shared](#) > [fShared](#)
- int32 [fMainIndex](#)
- uint32 [fIFDCount](#)
- [AutoPtr](#)< [dng_ifd](#) > [fIFD](#) [[kMaxSubIFDs](#)+1]
- uint32 [fChainedIFDCount](#)
- [AutoPtr](#)< [dng_ifd](#) > [fChainedIFD](#) [[kMaxChainedIFDs](#)]

Protected Member Functions

- virtual void [ValidateMagic](#) ()
- virtual void [ParseTag](#) ([dng_host](#) &host, [dng_stream](#) &stream, [dng_exif](#) *exif, [dng_shared](#) *shared, [dng_ifd](#) *ifd, uint32 parentCode, uint32 tagCode, uint32 tagType, uint32 tagCount, uint32 tagOffset, int32 offsetDelta)
- virtual bool [ValidateIFD](#) ([dng_stream](#) &stream, uint32 ifdOffset, int32 offsetDelta)

- virtual void **ParseIFD** ([dng_host](#) &host, [dng_stream](#) &stream, [dng_exif](#) *exif, [dng_shared](#) *shared, [dng_ifd](#) *ifd, uint32 ifdOffset, int32 offsetDelta, uint32 parentCode)
- virtual bool **ParseMakerNoteIFD** ([dng_host](#) &host, [dng_stream](#) &stream, uint32 ifdSize, uint32 ifdOffset, int32 offsetDelta, uint32 minOffset, uint32 maxOffset, uint32 parentCode)
- virtual void **ParseMakerNote** ([dng_host](#) &host, [dng_stream](#) &stream, uint32 makerNoteCount, uint32 makerNoteOffset, int32 offsetDelta, uint32 minOffset, uint32 maxOffset)
- virtual void **ParseSonyPrivateData** ([dng_host](#) &host, [dng_stream](#) &stream, uint32 count, uint32 oldOffset, uint32 newOffset)
- virtual void **ParseDNGPrivateData** ([dng_host](#) &host, [dng_stream](#) &stream)

Protected Attributes

- uint32 **fMakerNoteNextIFD**

6.30.1 Detailed Description

Top-level structure of DNG file with access to metadata.

See [DNG 1.1.0 specification](#) for information on member fields of this class.

6.30.2 Member Function Documentation

6.30.2.1 bool dng_info::IsValidDNG () [virtual]

Test validity of DNG data.

Return values:

true if stream provided a valid DNG.

6.30.2.2 void dng_info::Parse ([dng_host](#) & host, [dng_stream](#) & stream) [virtual]

Read [dng_info](#) from a [dng_stream](#)

Parameters:

host DNG host used for progress updating, abort testing, buffer allocation, etc.
stream Stream to read DNG data from.

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_info.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_info.cpp

6.31 dng_iptc Class Reference

Class for reading and holding IPTC metadata associated with a DNG file.

```
#include <dng_iptc.h>
```

Public Member Functions

- bool [IsEmpty](#) () const
- bool [NotEmpty](#) () const
- void [Parse](#) (const void *blockData, uint32 blockSize, uint32 offsetInOriginal-File)
- [dng_memory_block](#) * [Spool](#) ([dng_memory_allocator](#) &allocator)

Public Attributes

- [dng_string](#) **fTitle**
- int32 **fUrgency**
- [dng_string](#) **fCategory**
- [dng_string_list](#) **fSupplementalCategories**
- [dng_string_list](#) **fKeywords**
- [dng_string](#) **fInstructions**
- [dng_date_time](#) **fDateCreated**
- bool **fHasTimeCreated**
- int32 **fTimeZoneMinutes**
- [dng_string](#) **fAuthor**
- [dng_string](#) **fAuthorsPosition**
- [dng_string](#) **fCity**
- [dng_string](#) **fState**
- [dng_string](#) **fCountry**
- [dng_string](#) **fCountryCode**
- [dng_string](#) **fLocation**
- [dng_string](#) **fTransmissionReference**
- [dng_string](#) **fHeadline**
- [dng_string](#) **fCredit**
- [dng_string](#) **fSource**
- [dng_string](#) **fCopyrightNotice**
- [dng_string](#) **fDescription**
- [dng_string](#) **fDescriptionWriter**

Protected Types

- enum **DataSet** {
kRecordVersionSet = 0, **kObjectNameSet** = 5, **kUrgencySet** = 10, **kCategorySet** = 15,
kSupplementalCategoriesSet = 20, **kKeywordsSet** = 25, **kSpecialInstructionsSet** = 40, **kDateCreatedSet** = 55,
kTimeCreatedSet = 60, **kBylineSet** = 80, **kBylineTitleSet** = 85, **kCitySet** = 90,
kSublocationSet = 92, **kProvinceStateSet** = 95, **kCountryCodeSet** = 100, **kCountryNameSet** = 101,
kOriginalTransmissionReferenceSet = 103, **kHeadlineSet** = 105, **kCreditSet** = 110, **kSourceSet** = 115,
kCopyrightNoticeSet = 116, **kCaptionSet** = 120, **kCaptionWriterSet** = 122 }
- enum **CharSet** { **kCharSetUnknown** = 0, **kCharSetUTF8** = 1 }

Protected Member Functions

- void **ParseString** ([dng_stream](#) &stream, dng_string &s, CharSet charSet)
- void **SpoolString** ([dng_stream](#) &stream, const dng_string &s, uint8 dataSet, uint32 maxChars)

6.31.1 Detailed Description

Class for reading and holding IPTC metadata associated with a DNG file.

See the [IPTC specification](#) for information on member fields of this class.

6.31.2 Member Function Documentation

6.31.2.1 bool dng_iptc::IsEmpty () const

Test if IPTC metadata exists.

Return values:

true if no IPTC metadata exists for this DNG.

6.31.2.2 bool dng_iptc::NotEmpty () const [inline]

Test if IPTC metadata exists.

Return values:

true if IPTC metadata exists for this DNG.

6.31.2.3 void dng_iptc::Parse (const void * *blockData*, uint32 *blockSize*, uint32 *offsetInOriginalFile*)

Parse a complete block of IPTC data.

Parameters:

blockData The block of IPTC data.

blockSize Size in bytes of data block.

offsetInOriginalFile Used to enable certain file patching operations such as updating date/time in place.

6.31.2.4 dng_memory_block * dng_iptc::Spool (dng_memory_allocator & *allocator*)

Serialize IPTC data to a memory block.

Parameters:

allocator Memory allocator used to acquire memory block.

Return values:

Memory block

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_iptc.h
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_iptc.cpp

6.32 dng_linearization_info Class Reference

Class for managing data values related to DNG linearization.

```
#include <dng_linearization_info.h>
```

Public Member Functions

- void **RoundBlacks** ()
- virtual void **Parse** (dng_host &host, dng_stream &stream, dng_info &info)
- virtual void **PostParse** (dng_host &host, dng_negative &negative)
- real64 **MaxBlackLevel** (uint32 plane) const

Compute the maximum black level for a given sample plane taking into account base black level, repeated black level patter, and row/column delta maps.

- virtual void [Linearize](#) (dng_host &host, const dng_image &srcImage, dng_image &dstImage)
- dng_urational [BlackLevel](#) (uint32 row, uint32 col, uint32 plane) const
- uint32 [RowBlackCount](#) () const
Number of per-row black level deltas in fBlackDeltaV.
- dng_srrational [RowBlack](#) (uint32 row) const
- uint32 [ColumnBlackCount](#) () const
Number of per-column black level deltas in fBlackDeltaV.
- dng_srrational [ColumnBlack](#) (uint32 col) const

Public Attributes

- dng_rect [fActiveArea](#)
- uint32 [fMaskedAreaCount](#)
Number of rectangles in fMaskedArea.
- dng_rect [fMaskedArea](#) [kMaxMaskedAreas]
- [AutoPtr](#)< dng_memory_block > [fLinearizationTable](#)
- uint32 [fBlackLevelRepeatRows](#)
Actual number of rows in fBlackLevel pattern.
- uint32 [fBlackLevelRepeatCols](#)
Actual number of columns in fBlackLevel pattern.
- real64 [fBlackLevel](#) [kMaxBlackPattern][kMaxBlackPattern][kMaxSamplesPerPixel]
Repeating pattern of black level deltas fBlackLevelRepeatRows by fBlackLevelRepeatCols in size.
- [AutoPtr](#)< dng_memory_block > [fBlackDeltaH](#)
Memory block of double-precision floating point deltas between baseline black level and a given column's black level.
- [AutoPtr](#)< dng_memory_block > [fBlackDeltaV](#)
Memory block of double-precision floating point deltas between baseline black level and a given row's black level.
- real64 [fWhiteLevel](#) [kMaxSamplesPerPixel]
Single white level (maximum sensor value) for each sample plane.

Protected Attributes

- int32 **fBlackDenom**

6.32.1 Detailed Description

Class for managing data values related to DNG linearization.

See `LinearizationTable`, `BlackLevel`, `BlackLevelRepeatDim`, `BlackLevelDeltaH`, `BlackLevelDeltaV` and `WhiteLevel` tags in the [DNG 1.1.0 specification](#).

6.32.2 Member Function Documentation

6.32.2.1 `dng_urational dng_linearization_info::BlackLevel (uint32 row, uint32 col, uint32 plane) const`

Compute black level for one coordinate and sample plane in the image.

Parameters:

- row* Row to compute black level for.
- col* Column to compute black level for.
- plane* Sample plane to compute black level for.

6.32.2.2 `dng_srational dng_linearization_info::ColumnBlack (uint32 col) const`

Lookup black level delta for a given column.

Parameters:

- col* Column to get black level for.

Return values:

- black* level for indicated column.

6.32.2.3 `void dng_linearization_info::Linearize (dng_host & host, const dng_image & srcImage, dng_image & dstImage) [virtual]`

Convert raw data from in-file format to a true linear image using linearization data from DNG.

Parameters:

- host* Used to allocate buffers, check for aborts, and post progress updates.
- srcImage* Input pre-linearization RAW samples.
- dstImage* Output linearized image.

6.32.2.4 `dng_srational dng_linearization_info::RowBlack (uint32 row) const`

Lookup black level delta for a given row.

Parameters:

row Row to get black level for.

Return values:

black level for indicated row.

6.32.3 Member Data Documentation**6.32.3.1 `dng_rect dng_linearization_info::fActiveArea`**

This rectangle defines the active (non-masked) pixels of the sensor. The order of the rectangle coordinates is: top, left, bottom, right.

6.32.3.2 `AutoPtr<dng_memory_block> dng_linearization_info::fLinearizationTable`

A lookup table that maps stored values into linear values. This tag is typically used to increase compression ratios by storing the raw data in a non-linear, more visually uniform space with fewer total encoding levels. If SamplesPerPixel is not equal to one, e.g. Fuji S3 type sensor, this single table applies to all the samples for each pixel.

6.32.3.3 `dng_rect dng_linearization_info::fMaskedArea[kMaxMaskedAreas]`

List of non-overlapping rectangle coordinates of fully masked pixels. Can be optionally used by DNG readers to measure the black encoding level. The order of each rectangle's coordinates is: top, left, bottom, right. If the raw image data has already had its black encoding level subtracted, then this tag should not be used, since the masked pixels are no longer useful. Note that DNG writers are still required to include an estimate and store the black encoding level using the black level DNG tags. Support for the MaskedAreas tag is not required of DNG readers.

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_linearization_info.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_linearization_info.cpp

6.33 `dng_memory_allocator` Class Reference

Interface for [dng_memory_block](#) allocator.


```
#include <dng_memory.h>
```

Public Member Functions

- virtual [dng_memory_block](#) * [Allocate](#) (uint32 size)

6.33.1 Detailed Description

Interface for [dng_memory_block](#) allocator.

6.33.2 Member Function Documentation

6.33.2.1 [dng_memory_block](#) * [dng_memory_allocator::Allocate](#) (uint32 size)
[virtual]

Allocate a [dng_memory](#) block.

Parameters:

size Number of bytes in memory block.

Return values:

A [dng_memory_block](#) with at least size bytes of valid storage.

Exceptions:

[dng_exception](#) with `fErrorCode` equal to `dng_error_memory` .

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_memory.h
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_memory.cpp

6.34 dng_memory_block Class Reference

Class to provide resource acquisition is instantiation discipline for image buffers and other larger memory allocations.

```
#include <dng_memory.h>
```

Public Member Functions

- uint32 [LogicalSize](#) () const
- void * [Buffer](#) ()

- `const void * Buffer () const`
- `char * Buffer_char ()`
- `const char * Buffer_char () const`
- `uint8 * Buffer_uint8 ()`
- `const uint8 * Buffer_uint8 () const`
- `uint16 * Buffer_uint16 ()`
- `const uint16 * Buffer_uint16 () const`
- `int16 * Buffer_int16 ()`
- `const int16 * Buffer_int16 () const`
- `uint32 * Buffer_uint32 ()`
- `const uint32 * Buffer_uint32 () const`
- `int32 * Buffer_int32 ()`
- `const int32 * Buffer_int32 () const`
- `real32 * Buffer_real32 ()`
- `const real32 * Buffer_real32 () const`
- `real64 * Buffer_real64 ()`
- `const real64 * Buffer_real64 () const`

Protected Member Functions

- `dng_memory_block (uint32 logicalSize)`
- `uint32 PhysicalSize ()`
- `void SetBuffer (void *p)`

6.34.1 Detailed Description

Class to provide resource acquisition is instantiation discipline for image buffers and other larger memory allocations.

This class requires a [dng_memory_allocator](#) for allocation.

6.34.2 Member Function Documentation

6.34.2.1 `const void* dng_memory_block::Buffer () const` [inline]

Return pointer to allocated memory as a `const void *`.

Return values:

*const void ** valid for as many bytes as were allocated.

6.34.2.2 void* dng_memory_block::Buffer () [inline]

Return pointer to allocated memory as a void *.

Return values:

void * valid for as many bytes as were allocated.

6.34.2.3 const char* dng_memory_block::Buffer_char () const [inline]

Return pointer to allocated memory as a const char *.

Return values:

const char * valid for as many bytes as were allocated.

6.34.2.4 char* dng_memory_block::Buffer_char () [inline]

Return pointer to allocated memory as a char *.

Return values:

char * valid for as many bytes as were allocated.

6.34.2.5 const int16* dng_memory_block::Buffer_int16 () const [inline]

Return pointer to allocated memory as a const int16 *.

Return values:

const int16 * valid for as many bytes as were allocated.

6.34.2.6 int16* dng_memory_block::Buffer_int16 () [inline]

Return pointer to allocated memory as a int16 *.

Return values:

int16 * valid for as many bytes as were allocated.

6.34.2.7 const int32* dng_memory_block::Buffer_int32 () const [inline]

Return pointer to allocated memory as a const int32 *.

Return values:

const int32 * valid for as many bytes as were allocated.

6.34.2.8 int32* dng_memory_block::Buffer_int32 () [inline]

Return pointer to allocated memory as a int32 *.

Return values:

int32 * valid for as many bytes as were allocated.

6.34.2.9 const real32* dng_memory_block::Buffer_real32 () const [inline]

Return pointer to allocated memory as a const real32 *.

Return values:

const real32 * valid for as many bytes as were allocated.

6.34.2.10 real32* dng_memory_block::Buffer_real32 () [inline]

Return pointer to allocated memory as a real32 *.

Return values:

real32 * valid for as many bytes as were allocated.

6.34.2.11 const real64* dng_memory_block::Buffer_real64 () const [inline]

Return pointer to allocated memory as a const real64 *.

Return values:

const real64 * valid for as many bytes as were allocated.

6.34.2.12 real64* dng_memory_block::Buffer_real64 () [inline]

Return pointer to allocated memory as a real64 *.

Return values:

real64 * valid for as many bytes as were allocated.

6.34.2.13 const uint16* dng_memory_block::Buffer_uint16 () const [inline]

Return pointer to allocated memory as a const uint16 *.

Return values:

const uint16 * valid for as many bytes as were allocated.

6.34.2.14 uint16* dng_memory_block::Buffer_uint16 () [inline]

Return pointer to allocated memory as a uint16 *.

Return values:

uint16 * valid for as many bytes as were allocated.

6.34.2.15 const uint32* dng_memory_block::Buffer_uint32 () const [inline]

Return pointer to allocated memory as a const uint32 *.

Return values:

const uint32 * valid for as many bytes as were allocated.

6.34.2.16 uint32* dng_memory_block::Buffer_uint32 () [inline]

Return pointer to allocated memory as a uint32 *.

Return values:

uint32 * valid for as many bytes as were allocated.

6.34.2.17 const uint8* dng_memory_block::Buffer_uint8 () const [inline]

Return pointer to allocated memory as a const uint8 *.

Return values:

const uint8 * valid for as many bytes as were allocated.

6.34.2.18 uint8* dng_memory_block::Buffer_uint8 () [inline]

Return pointer to allocated memory as a uint8 *.

Return values:

uint8 * valid for as many bytes as were allocated.

6.34.2.19 uint32 dng_memory_block::LogicalSize () const [inline]

Getter for available size, in bytes, of memory block.

Return values:

size in bytes of available memory in memory block.

The documentation for this class was generated from the following file:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_memory.h

6.35 dng_memory_data Class Reference

Class to provide resource acquisition is instantiation discipline for small memory allocations.

```
#include <dng_memory.h>
```

Public Member Functions

- [dng_memory_data](#) ()
- [dng_memory_data](#) (uint32 size)
- [~dng_memory_data](#) ()
Release memory buffer using free.
- void [Allocate](#) (uint32 size)
- void [Clear](#) ()
Release any allocated memory using free. Object is still valid and Allocate can be called again.
- void * [Buffer](#) ()
- const void * [Buffer](#) () const
- char * [Buffer_char](#) ()
- const char * [Buffer_char](#) () const
- uint8 * [Buffer_uint8](#) ()
- const uint8 * [Buffer_uint8](#) () const
- uint16 * [Buffer_uint16](#) ()
- const uint16 * [Buffer_uint16](#) () const
- int16 * [Buffer_int16](#) ()
- const int16 * [Buffer_int16](#) () const
- uint32 * [Buffer_uint32](#) ()
- const uint32 * [Buffer_uint32](#) () const
- int32 * [Buffer_int32](#) ()
- const int32 * [Buffer_int32](#) () const
- real32 * [Buffer_real32](#) ()
- const real32 * [Buffer_real32](#) () const
- real64 * [Buffer_real64](#) ()
- const real64 * [Buffer_real64](#) () const

6.35.1 Detailed Description

Class to provide resource acquisition is instantiation discipline for small memory allocations.

This class does not use [dng_memory_allocator](#) for memory allocation.

6.35.2 Constructor & Destructor Documentation

6.35.2.1 dng_memory_data::dng_memory_data ()

Construct an empty memory buffer using malloc.

Exceptions:

dng_memory_full with fErrorCode equal to dng_error_memory.

6.35.2.2 dng_memory_data::dng_memory_data (uint32 size)

Construct memory buffer of size bytes using malloc.

Parameters:

size Number of bytes of memory needed.

Exceptions:

dng_memory_full with fErrorCode equal to dng_error_memory.

6.35.3 Member Function Documentation

6.35.3.1 void dng_memory_data::Allocate (uint32 size)

Clear existing memory buffer and allocate new memory of size bytes.

Parameters:

size Number of bytes of memory needed.

Exceptions:

dng_memory_full with fErrorCode equal to dng_error_memory.

6.35.3.2 const void* dng_memory_data::Buffer () const [inline]

Return pointer to allocated memory as a const void *.

Return values:

const void * valid for as many bytes as were allocated.

6.35.3.3 void* dng_memory_data::Buffer () [inline]

Return pointer to allocated memory as a void *.

Return values:

void * valid for as many bytes as were allocated.

6.35.3.4 const char* dng_memory_data::Buffer_char () const [inline]

Return pointer to allocated memory as a const char *.

Return values:

const char * valid for as many bytes as were allocated.

6.35.3.5 char* dng_memory_data::Buffer_char () [inline]

Return pointer to allocated memory as a char *.

Return values:

char * valid for as many bytes as were allocated.

6.35.3.6 const int16* dng_memory_data::Buffer_int16 () const [inline]

Return pointer to allocated memory as a const int16 *.

Return values:

const int16 * valid for as many bytes as were allocated.

6.35.3.7 int16* dng_memory_data::Buffer_int16 () [inline]

Return pointer to allocated memory as a int16 *.

Return values:

int16 * valid for as many bytes as were allocated.

6.35.3.8 const int32* dng_memory_data::Buffer_int32 () const [inline]

Return pointer to allocated memory as a const int32 *.

Return values:

const int32 * valid for as many bytes as were allocated.

6.35.3.9 int32* dng_memory_data::Buffer_int32 () [inline]

Return pointer to allocated memory as a const int32 *.

Return values:

const int32 * valid for as many bytes as were allocated.

6.35.3.10 const real32* dng_memory_data::Buffer_real32 () const [inline]

Return pointer to allocated memory as a const real32 *.

Return values:

const real32 * valid for as many bytes as were allocated.

6.35.3.11 real32* dng_memory_data::Buffer_real32 () [inline]

Return pointer to allocated memory as a real32 *.

Return values:

real32 * valid for as many bytes as were allocated.

6.35.3.12 const real64* dng_memory_data::Buffer_real64 () const [inline]

Return pointer to allocated memory as a const real64 *.

Return values:

const real64 * valid for as many bytes as were allocated.

6.35.3.13 real64* dng_memory_data::Buffer_real64 () [inline]

Return pointer to allocated memory as a real64 *.

Return values:

real64 * valid for as many bytes as were allocated.

6.35.3.14 const uint16* dng_memory_data::Buffer_uint16 () const [inline]

Return pointer to allocated memory as a const uint16 *.

Return values:

const uint16 * valid for as many bytes as were allocated.

6.35.3.15 uint16* dng_memory_data::Buffer_uint16 () [inline]

Return pointer to allocated memory as a uint16 *.

Return values:

uint16 * valid for as many bytes as were allocated.

6.35.3.16 const uint32* dng_memory_data::Buffer_uint32 () const [inline]

Return pointer to allocated memory as a uint32 *.

Return values:

uint32 * valid for as many bytes as were allocated.

6.35.3.17 uint32* dng_memory_data::Buffer_uint32 () [inline]

Return pointer to allocated memory as a uint32 *.

Return values:

uint32 * valid for as many bytes as were allocated.

6.35.3.18 const uint8* dng_memory_data::Buffer_uint8 () const [inline]

Return pointer to allocated memory as a const uint8 *.

Return values:

const uint8 * valid for as many bytes as were allocated.

6.35.3.19 uint8* dng_memory_data::Buffer_uint8 () [inline]

Return pointer to allocated memory as a uint8 *.

Return values:

uint8 * valid for as many bytes as were allocated.

The documentation for this class was generated from the following files:

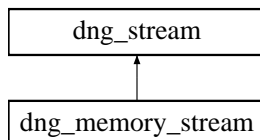
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_memory.h
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_memory.cpp

6.36 dng_memory_stream Class Reference

A [dng_stream](#) which can be read from or written to memory.

```
#include <dng_memory_stream.h>
```

Inheritance diagram for dng_memory_stream::



Public Member Functions

- [dng_memory_stream](#) ([dng_memory_allocator](#) &allocator, [dng_abort_sniffer](#) *sniffer=NULL, uint32 pageSize=64 *1024)

Protected Member Functions

- virtual uint32 **DoGetLength** ()
- virtual void **DoRead** (void *data, uint32 count, uint32 offset)
- virtual void **DoSetLength** (uint32 length)
- virtual void **DoWrite** (const void *data, uint32 count, uint32 offset)

Protected Attributes

- [dng_memory_allocator](#) & fAllocator
- uint32 fPageSize
- uint32 fPageCount
- uint32 fPagesAllocated
- [dng_memory_block](#) ** fPageList
- uint32 fMemoryStreamLength

6.36.1 Detailed Description

A [dng_stream](#) which can be read from or written to memory.

Stream is populated via writing and either read or accessed by asking for contents as a pointer.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 dng_memory_stream::dng_memory_stream (dng_memory_allocator & allocator, dng_abort_sniffer * sniffer = NULL, uint32 pageSize = 64 * 1024)

Construct a new memory-based stream.

Parameters:

allocator Allocator to use to allocate memory in stream as needed.

sniffer If non-NULL used to check for user cancellation.

pageSize Unit of allocation for data stored in stream.

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_memory_stream.h
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_memory_stream.cpp

6.37 dng_mosaic_info Class Reference

Support for describing color filter array patterns and manipulating mosaic sample data.

```
#include <dng_mosaic_info.h>
```

Public Member Functions

- virtual void **Parse** (dng_host &host, dng_stream &stream, dng_info &info)
- virtual void **PostParse** (dng_host &host, dng_negative &negative)
- bool **IsColorFilterArray** () const
- virtual bool **SetFourColorBayer** ()
- virtual dng_point **FullScale** () const
- virtual dng_point **DownScale** (uint32 minSize) const
- virtual dng_point **DstSize** (const dng_point &downScale) const
- virtual void **InterpolateGeneric** (dng_host &host, const dng_image &srcImage, dng_image &dstImage, uint32 srcPlane=0) const
- virtual void **InterpolateFast** (dng_host &host, const dng_image &srcImage, dng_image &dstImage, const dng_point &downScale, uint32 srcPlane=0) const
- virtual void **Interpolate** (dng_host &host, const dng_image &srcImage, dng_image &dstImage, const dng_point &downScale, uint32 srcPlane=0) const

Public Attributes

- dng_point [fCFAPatternSize](#)
Size of fCFAPattern.
- uint8 [fCFAPattern](#) [[kMaxCFAPattern](#)][[kMaxCFAPattern](#)]
CFA pattern from CFAPattern tag in the [TIFF/EP specification](#)..
- uint32 [fColorPlanes](#)
Number of color planes in DNG input.
- uint8 [fCFAPlaneColor](#) [[kMaxColorPlanes](#)]
- uint32 [fCFALayout](#)
- uint32 [fBayerGreenSplit](#)

Protected Member Functions

- virtual bool **IsSafeDownScale** (const dng_point &downScale) const
- virtual bool **ValidSizeDownScale** (const dng_point &downScale, uint32 minSize) const

Protected Attributes

- dng_point [fSrcSize](#)
- dng_point [fCroppedSize](#)
- real64 [fAspectRatio](#)

6.37.1 Detailed Description

Support for describing color filter array patterns and manipulating mosaic sample data. See CFAPattern tag in [TIFF/EP specification](#) and CFAPlaneColor, CFALayout, and BayerGreenSplit tags in the [DNG 1.1.0 specification](#).

6.37.2 Member Function Documentation

6.37.2.1 dng_point dng_mosaic_info::DownScale (uint32 minSize) const [virtual]

Returns integer factors by which mosaic data must be downsampled to produce an image which is as close to minSize as possible in longer dimension, but no smaller.

Parameters:

minSize Number of pixels as target for longer dimension of downsampled image.

Return values:

Point containing integer factors by which image must be downsampled.

6.37.2.2 dng_point dng_mosaic_info::DstSize (const dng_point & downScale) const [virtual]

Return size of demosaiced image for passed in downscaling factor.

Parameters:

downScale Integer downsampling factor obtained from DownScale method.

Return values:

Size of resulting demosaiced image.

6.37.2.3 dng_point dng_mosaic_info::FullScale () const [virtual]

Returns scaling factor relative to input size needed to capture output data. Staggered (or rotated) sensing arrays are produced to a larger output than the number of input samples. This method indicates how much larger.

Return values:

a point with integer scaling factors for the horizontal and vertical dimensions.

6.37.2.4 void dng_mosaic_info::Interpolate (dng_host & host, const dng_image & srcImage, dng_image & dstImage, const dng_point & downScale, uint32 srcPlane = 0) const [virtual]

Demosaic interpolation of a single plane. Chooses between generic and fast interpolators based on parameters.

Parameters:

host [dng_host](#) to use for buffer allocation requests, user cancellation testing, and progress updates.

srcImage Source image for mosaiced data.

dstImage Destination image for resulting interpolated data.

downScale Amount (in horizontal and vertical) by which to subsample image.

srcPlane Which plane to interpolate.

6.37.2.5 void dng_mosaic_info::InterpolateFast (dng_host & host, const dng_image & srcImage, dng_image & dstImage, const dng_point & downScale, uint32 srcPlane = 0) const [virtual]

Demosaic interpolation of a single plane for downsampled case.

Parameters:

host dng_host to use for buffer allocation requests, user cancellation testing, and progress updates.

srcImage Source image for mosaiced data.

dstImage Destination image for resulting interpolated data.

downScale Amount (in horizontal and vertical) by which to subsample image.

srcPlane Which plane to interpolate.

6.37.2.6 void dng_mosaic_info::InterpolateGeneric (dng_host & host, const dng_image & srcImage, dng_image & dstImage, uint32 srcPlane = 0) const [virtual]

Demosaic interpolation of a single plane for non-downsampled case.

Parameters:

host dng_host to use for buffer allocation requests, user cancellation testing, and progress updates.

srcImage Source image for mosaiced data.

dstImage Destination image for resulting interpolated data.

srcPlane Which plane to interpolate.

6.37.2.7 bool dng_mosaic_info::IsColorFilterArray () const [inline]

Returns whether the RAW data in this DNG file from a color filter array (mosaiced) source.

Return values:

true if this DNG file is from a color filter array (mosaiced) source.

6.37.2.8 bool dng_mosaic_info::SetFourColorBayer () [virtual]

Enable generating four-plane output from three-plane Bayer input. Extra plane is a second version of the green channel. First green is produced using green mosaic samples from one set of rows/columns (even/odd) and the second green channel is produced using the other set of rows/columns. One can compare the two versions to judge whether BayerGreenSplit needs to be set for a given input source.

6.37.3 Member Data Documentation

6.37.3.1 uint32 [dng_mosaic_info::fBayerGreenSplit](#)

Value of BayerGreenSplit tag in DNG file. BayerGreenSplit only applies to CFA images using a Bayer pattern filter array. This tag specifies, in arbitrary units, how closely the values of the green pixels in the blue/green rows track the values of the green pixels in the red/green rows.

A value of zero means the two kinds of green pixels track closely, while a non-zero value means they sometimes diverge. The useful range for this tag is from 0 (no divergence) to about 5000 (large divergence).

6.37.3.2 uint32 [dng_mosaic_info::fCFALayout](#)

Value of CFALayout tag in the [DNG 1.1.0 specification](#). CFALayout describes the spatial layout of the CFA. The currently defined values are:

- 1 = Rectangular (or square) layout.
- 2 = Staggered layout A: even columns are offset down by 1/2 row.
- 3 = Staggered layout B: even columns are offset up by 1/2 row.
- 4 = Staggered layout C: even rows are offset right by 1/2 column.
- 5 = Staggered layout D: even rows are offset left by 1/2 column.

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_mosaic_info.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_mosaic_info.cpp

6.38 dng_negative Class Reference

Main class for holding DNG image data and associated metadata.

```
#include <dng_negative.h>
```

Public Member Functions

- [dng_memory_allocator](#) & [Allocator](#) () const
Provide access to the memory allocator used for this object.

- void [SetModelName](#) (const char *name)
Getter for ModelName.
- const dng_string & [ModelName](#) () const
Setter for ModelName.
- void [SetLocalName](#) (const char *name)
Setter for LocalName.
- const dng_string & [LocalName](#) () const
Getter for LocalName.
- void [SetBaseOrientation](#) (const dng_orientation &orientation)
Setter for BaseOrientation.
- bool [HasBaseOrientation](#) () const
Has BaseOrientation been set?
- const dng_orientation & [BaseOrientation](#) () const
Getter for BaseOrientation.
- virtual dng_orientation [Orientation](#) () const
Hook to allow SDK host code to add additional rotations.
- void [ApplyOrientation](#) (const dng_orientation &orientation)
- void [SetDefaultCropSize](#) (const dng_urational &sizeH, const dng_urational &sizeV)
Setter for DefaultCropSize.
- void [SetDefaultCropSize](#) (uint32 sizeH, uint32 sizeV)
Setter for DefaultCropSize.
- const dng_urational & [DefaultCropSizeH](#) () const
Getter for DefaultCropSize horizontal.
- const dng_urational & [DefaultCropSizeV](#) () const
Getter for DefaultCropSize vertical.
- void [SetDefaultCropOrigin](#) (const dng_urational &originH, const dng_urational &originV)
Setter for DefaultCropOrigin.
- void [SetDefaultCropOrigin](#) (uint32 originH, uint32 originV)

Setter for DefaultCropOrigin.

- void [SetDefaultCropCentered](#) (const dng_point &rawSize)
Set default crop around center of image.
- const dng_urational & [DefaultCropOriginH](#) () const
Get default crop origin horizontal value.
- const dng_urational & [DefaultCropOriginV](#) () const
Get default crop origin vertical value.
- void [SetDefaultScale](#) (const dng_urational &scaleH, const dng_urational &scaleV)
Setter for DefaultScale.
- const dng_urational & [DefaultScaleH](#) () const
Get default scale horizontal value.
- const dng_urational & [DefaultScaleV](#) () const
Get default scale vertical value.
- void [SetBestQualityScale](#) (const dng_urational &scale)
Setter for BestQualityScale.
- const dng_urational & [BestQualityScale](#) () const
Getter for BestQualityScale.
- real64 [RawToFullScaleH](#) () const
API for raw to full image scaling factors horizontal.
- real64 [RawToFullScaleV](#) () const
API for raw to full image scaling factors vertical.
- real64 [DefaultScale](#) () const
- real64 [SquareWidth](#) () const
Default cropped image size (at scale == 1.0) width.
- real64 [SquareHeight](#) () const
Default cropped image size (at scale == 1.0) height.
- real64 [AspectRatio](#) () const
Default cropped image aspect ratio.

- uint32 [FinalWidth](#) (real64 scale) const
Default cropped image size at given scale factor width.
- uint32 [FinalHeight](#) (real64 scale) const
Default cropped image size at given scale factor height.
- uint32 [DefaultFinalWidth](#) () const
Default cropped image size at default scale factor width.
- uint32 [DefaultFinalHeight](#) () const
Default cropped image size at default scale factor height.
- uint32 [BestQualityFinalWidth](#) () const
- uint32 [BestQualityFinalHeight](#) () const
- dng_rect [DefaultCropArea](#) (real64 scaleH=1.0, real64 scaleV=1.0) const
The default crop area after applying the specified horizontal and vertical scale factors to the stage 3 image.
- void [SetBaselineNoise](#) (real64 noise)
Setter for BaselineNoise.
- const dng_urational & [BaselineNoiseR](#) () const
Getter for BaselineNoise as dng_urational.
- real64 [BaselineNoise](#) () const
Getter for BaselineNoise as real64.
- void [SetBaselineExposure](#) (real64 exposure)
Setter for BaselineExposure.
- const dng_srational & [BaselineExposureR](#) () const
Getter for BaselineExposure as dng_urational.
- real64 [BaselineExposure](#) () const
Getter for BaselineExposure as real64.
- void [SetBaselineSharpness](#) (real64 sharpness)
Setter for BaselineSharpness.
- const dng_urational & [BaselineSharpnessR](#) () const
Getter for BaselineSharpness as dng_urational.
- real64 [BaselineSharpness](#) () const

Getter for BaselineSharpness as real64.

- void [SetChromaBlurRadius](#) (const dng_urational &radius)
Setter for ChromaBlurRadius.
- const dng_urational & [ChromaBlurRadius](#) () const
Getter for ChromaBlurRadius as dng_urational.
- void [SetAntiAliasStrength](#) (const dng_urational &strength)
Setter for AntiAliasStrength.
- const dng_urational & [AntiAliasStrength](#) () const
Getter for AntiAliasStrength as dng_urational.
- void [SetLinearResponseLimit](#) (real64 limit)
Setter for LinearResponseLimit.
- const dng_urational & [LinearResponseLimitR](#) () const
Getter for LinearResponseLimit as dng_urational.
- real64 [LinearResponseLimit](#) () const
Getter for LinearResponseLimit as real64.
- void [SetShadowScale](#) (const dng_urational &scale)
Setter for ShadowScale.
- const dng_urational & [ShadowScaleR](#) () const
Getter for ShadowScale as dng_urational.
- real64 [ShadowScale](#) () const
Getter for ShadowScale as real64.
- void **SetColorimetricReference** (uint32 ref)
- uint32 **ColorimetricReference** () const
- void [SetColorChannels](#) (uint32 channels)
Setter for ColorChannels.
- uint32 [ColorChannels](#) () const
Getter for ColorChannels.
- void [SetMonochrome](#) ()
Setter for MonoChrome.

- bool [IsMonochrome](#) () const
Getter for MonoChrome.
- void [SetAnalogBalance](#) (const dng_vector &b)
Setter for AnalogBalance.
- dng_urational [AnalogBalanceR](#) (uint32 channel) const
Getter for AnalogBalance as dng_urational.
- real64 [AnalogBalance](#) (uint32 channel) const
Getter for AnalogBalance as real64.
- void [SetCameraNeutral](#) (const dng_vector &n)
Setter for CameraNeutral.
- void [ClearCameraNeutral](#) ()
Clear CameraNeutral.
- bool [HasCameraNeutral](#) () const
Determine if CameraNeutral has been set but not cleared.
- const dng_vector & [CameraNeutral](#) () const
Getter for CameraNeutral.
- dng_urational [CameraNeutralR](#) (uint32 channel) const
- void [SetCameraWhiteXY](#) (const dng_xy_coord &coord)
Setter for CameraWhiteXY.
- bool [HasCameraWhiteXY](#) () const
- const dng_xy_coord & [CameraWhiteXY](#) () const
- void [GetCameraWhiteXY](#) (dng_urational &x, dng_urational &y) const
- void [SetEmbeddedCameraProfile](#) ([AutoPtr](#)< [dng_camera_profile](#) > &profile)
- void [ClearEmbeddedCameraProfile](#) ()
- const [dng_camera_profile](#) * [EmbeddedCameraProfile](#) () const
- virtual const [dng_camera_profile](#) * [CameraProfileToEmbed](#) () const
- virtual [dng_color_spec](#) * [MakeColorSpec](#) (const char *name=NULL) const
- void [SetRawDataUniqueID](#) (const [dng_fingerprint](#) &id)
- const [dng_fingerprint](#) & [RawDataUniqueID](#) () const
- void [FindRawDataUniqueID](#) ([dng_host](#) &host)
- void [SetOriginalRawFileName](#) (const char *name)
- bool [HasOriginalRawFileName](#) () const
- const dng_string & [OriginalRawFileName](#) () const

- void **SetHasOriginalRawFileData** (bool hasData)
- bool **CanEmbedOriginalRaw** () const
- void **SetOriginalRawFileData** (AutoPtr< dng_memory_block > &data)
- const void * **OriginalRawFileData** () const
- uint32 **OriginalRawFileDataLength** () const
- void **SetPrivateData** (AutoPtr< dng_memory_block > &block)
- void **ClearPrivateData** ()
- const uint8 * **PrivateData** () const
- uint32 **PrivateLength** () const
- void **SetMakerNoteSafety** (bool safe)
- bool **IsMakerNoteSafe** () const
- void **SetMakerNote** (AutoPtr< dng_memory_block > &block)
- void **ClearMakerNote** ()
- const void * **MakerNoteData** () const
- uint32 **MakerNoteLength** () const
- dng_exif * **GetExif** ()
- const dng_exif * **GetExif** () const
- virtual dng_memory_block * **BuildExifBlock** (const dng_resolution *resolution=NULL, bool includeIPTC=false) const
- void **SetIPTC** (AutoPtr< dng_memory_block > &block, uint32 offset)
- void **SetIPTC** (AutoPtr< dng_memory_block > &block)
- void **ClearIPTC** ()
- const void * **IPTCData** () const
- uint32 **IPTCLength** () const
- uint32 **IPTCOffset** () const
- dng_fingerprint **IPTCDigest** () const
- void **RebuildIPTC** ()
- bool **SetXMP** (dng_host &host, const void *buffer, uint32 count, bool xmpInSidecar=false, bool xmpIsNewer=false)
- dng_xmp * **GetXMP** ()
- const dng_xmp * **GetXMP** () const
- bool **XMPinSidecar** () const
- const dng_linearization_info * **GetLinearizationInfo** () const
- void **ClearLinearizationInfo** ()
- void **SetLinearization** (AutoPtr< dng_memory_block > &curve)
- void **SetActiveArea** (const dng_rect &area)
- void **SetMaskedAreas** (uint32 count, const dng_rect *area)
- void **SetMaskedArea** (const dng_rect &area)
- void **SetBlackLevel** (real64 black, int32 plane=-1)
- void **SetQuadBlacks** (real64 black0, real64 black1, real64 black2, real64 black3)
- void **SetRowBlacks** (const real64 *blacks, uint32 count)
- void **SetColumnBlacks** (const real64 *blacks, uint32 count)

- uint32 **WhiteLevel** (uint32 plane=0) const
- void **SetWhiteLevel** (uint32 white, int32 plane=-1)
- const [dng_mosaic_info](#) * **GetMosaicInfo** () const
- void **ClearMosaicInfo** ()
- void **SetColorKeys** (ColorKeyCode color0, ColorKeyCode color1, ColorKeyCode color2, ColorKeyCode color3=colorKeyMaxEnum)
- void **SetRGB** ()
- void **SetCMY** ()
- void **SetGMCY** ()
- void **SetBayerMosaic** (uint32 phase)
- void **SetFujiMosaic** (uint32 phase)
- void **SetQuadMosaic** (uint32 pattern)
- void **SetGreenSplit** (uint32 split)
- virtual void **Parse** ([dng_host](#) &host, [dng_stream](#) &stream, [dng_info](#) &info)
- virtual void **PostParse** ([dng_host](#) &host, [dng_stream](#) &stream, [dng_info](#) &info)

- virtual void **SynchronizeMetadata** ()
- void **UpdateDateTime** (const [dng_date_time](#) &dt, int32 tzHour=0x7FFFFFFF)
- void **UpdateDateTimeToNow** ()
- virtual bool **SetFourColorBayer** ()
- const [dng_image](#) * **Stage1Image** () const
- const [dng_image](#) * **Stage2Image** () const
- const [dng_image](#) * **Stage3Image** () const
- const [dng_image](#) & **RawImage** () const
- virtual [dng_image](#) * **MakeImage** (const [dng_rect](#) &bounds, uint32 planes, uint32 pixelType, uint32 pixelRange) const
- virtual void **ReadStage1Image** ([dng_host](#) &host, [dng_stream](#) &stream, [dng_info](#) &info)
- void **SetStage1Image** ([AutoPtr](#)< [dng_image](#) > &image)
- virtual void **BuildStage2Image** ([dng_host](#) &host, uint32 pixelType=ttShort)
- virtual void **ClearStage1** ()
- virtual void **BuildStage3Image** ([dng_host](#) &host, int32 srcPlane=-1)
- real64 **Stage3Gain** () const
- virtual void **ClearStage2** ()
- virtual void **ClearStage3** ()
- void **SetIsPreview** (bool preview)
- bool **IsPreview** () const

Static Public Member Functions

- static [dng_negative](#) * **Make** ([dng_memory_allocator](#) &allocator)

Protected Member Functions

- **dng_negative** ([dng_memory_allocator](#) &allocator)
- virtual void **Initialize** ()
- virtual [dng_exif](#) * **MakeExif** ()
- virtual [dng_xmp](#) * **MakeXMP** ()
- virtual [dng_linearization_info](#) * **MakeLinearizationInfo** ()
- void **NeedLinearizationInfo** ()
- virtual [dng_mosaic_info](#) * **MakeMosaicInfo** ()
- void **NeedMosaicInfo** ()
- virtual uint16 **LinearRange16** () const
- virtual void **DoBuildStage2** ([dng_host](#) &host, uint32 pixelType)
- virtual void **DoBuildStage3** ([dng_host](#) &host, uint32 minSize, int32 srcPlane)
- virtual void **DoMergeStage3** ([dng_host](#) &host)

Protected Attributes

- [dng_memory_allocator](#) & **fAllocator**
- [dng_string](#) **fModelName**
- [dng_string](#) **fLocalName**
- bool **fHasBaseOrientation**
- [dng_orientation](#) **fBaseOrientation**
- [dng_urational](#) **fDefaultCropSizeH**
- [dng_urational](#) **fDefaultCropSizeV**
- [dng_urational](#) **fDefaultCropOriginH**
- [dng_urational](#) **fDefaultCropOriginV**
- [dng_urational](#) **fDefaultScaleH**
- [dng_urational](#) **fDefaultScaleV**
- [dng_urational](#) **fBestQualityScale**
- real64 **fRawToFullScaleH**
- real64 **fRawToFullScaleV**
- [dng_urational](#) **fBaselineNoise**
- [dng_srational](#) **fBaselineExposure**
- [dng_urational](#) **fBaselineSharpness**
- [dng_urational](#) **fChromaBlurRadius**
- [dng_urational](#) **fAntiAliasStrength**
- [dng_urational](#) **fLinearResponseLimit**
- [dng_urational](#) **fShadowScale**
- uint32 **fColorimetricReference**
- uint32 **fColorChannels**
- [dng_vector](#) **fAnalogBalance**
- [dng_vector](#) **fCameraNeutral**
- [dng_xy_coord](#) **fCameraWhiteXY**

- [AutoPtr< dng_camera_profile >](#) **fEmbeddedCameraProfile**
- [dng_fingerprint](#) **fRawDataUniqueID**
- [dng_string](#) **fOriginalRawFileName**
- **bool fHasOriginalRawFileData**
- [AutoPtr< dng_memory_block >](#) **fOriginalRawFileData**
- [AutoPtr< dng_memory_block >](#) **fDNGPrivateData**
- **bool fIsMakerNoteSafe**
- [AutoPtr< dng_memory_block >](#) **fMakerNote**
- [AutoPtr< dng_exif >](#) **fExif**
- [AutoPtr< dng_memory_block >](#) **fIPTCBlock**
- **uint32 fIPTCOffset**
- [AutoPtr< dng_xmp >](#) **fXMP**
- **bool fXMPinSidecar**
- **bool fXMPisNewer**
- [AutoPtr< dng_linearization_info >](#) **fLinearizationInfo**
- [AutoPtr< dng_mosaic_info >](#) **fMosaicInfo**
- [AutoPtr< dng_image >](#) **fStage1Image**
- [AutoPtr< dng_image >](#) **fStage2Image**
- [AutoPtr< dng_image >](#) **fStage3Image**
- **real64 fStage3Gain**
- **bool fIsPreview**

6.38.1 Detailed Description

Main class for holding DNG image data and associated metadata.

6.38.2 Member Function Documentation

6.38.2.1 void dng_negative::ApplyOrientation (const dng_orientation & *orientation*)

Logically rotates the image by changing the orientation values. This will also update the XMP data.

6.38.2.2 uint32 dng_negative::BestQualityFinalHeight () const [inline]

Get best quality height. For a naive conversion, one could use either the default size, or the best quality size.

6.38.2.3 uint32 dng_negative::BestQualityFinalWidth () const [inline]

Get best quality width. For a naive conversion, one could use either the default size, or the best quality size.

6.38.2.4 real64 dng_negative::DefaultScale () const [inline]

Get default scale factor. When specifying a single scale factor, we use the horizontal scale factor, and let the vertical scale factor be calculated based on the pixel aspect ratio.

The documentation for this class was generated from the following files:

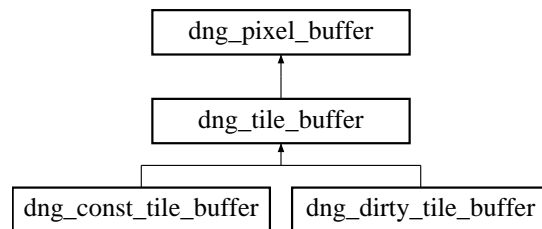
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_negative.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_negative.cpp

6.39 dng_pixel_buffer Class Reference

Holds a buffer of pixel data with "pixel geometry" metadata.

```
#include <dng_pixel_buffer.h>
```

Inheritance diagram for dng_pixel_buffer::

**Public Member Functions**

- **dng_pixel_buffer** (const [dng_pixel_buffer](#) &buffer)
- [dng_pixel_buffer](#) & **operator=** (const [dng_pixel_buffer](#) &buffer)
- uint32 [PixelRange](#) () const
- const void * [ConstPixel](#) (int32 row, int32 col, uint32 plane=0) const
- void * [DirtyPixel](#) (int32 row, int32 col, uint32 plane=0)
- const uint8 * [ConstPixel_uint8](#) (int32 row, int32 col, uint32 plane=0) const
- uint8 * [DirtyPixel_uint8](#) (int32 row, int32 col, uint32 plane=0)
- const int8 * [ConstPixel_int8](#) (int32 row, int32 col, uint32 plane=0) const
- int8 * [DirtyPixel_int8](#) (int32 row, int32 col, uint32 plane=0)
- const uint16 * [ConstPixel_uint16](#) (int32 row, int32 col, uint32 plane=0) const
- uint16 * [DirtyPixel_uint16](#) (int32 row, int32 col, uint32 plane=0)
- const int16 * [ConstPixel_int16](#) (int32 row, int32 col, uint32 plane=0) const
- int16 * [DirtyPixel_int16](#) (int32 row, int32 col, uint32 plane=0)
- const uint32 * [ConstPixel_uint32](#) (int32 row, int32 col, uint32 plane=0) const

- uint32 * [DirtyPixel_uint32](#) (int32 row, int32 col, uint32 plane=0)
- const int32 * [ConstPixel_int32](#) (int32 row, int32 col, uint32 plane=0) const
- int32 * [DirtyPixel_int32](#) (int32 row, int32 col, uint32 plane=0)
- const real32 * [ConstPixel_real32](#) (int32 row, int32 col, uint32 plane=0) const
- real32 * [DirtyPixel_real32](#) (int32 row, int32 col, uint32 plane=0)
- void [SetZero](#) (const dng_rect &area, uint32 plane, uint32 planes)
- void [CopyArea](#) (const [dng_pixel_buffer](#) &src, const dng_rect &area, uint32 plane, uint32 planes)
- void [RepeatArea](#) (const dng_rect &srcArea, const dng_rect &dstArea)
- void [ShiftRight](#) (uint32 shift)
- void [FlipH](#) ()
- void [FlipV](#) ()
- void [FlipZ](#) ()

Static Public Member Functions

- static dng_point [RepeatPhase](#) (const dng_rect &srcArea, const dng_rect &dstArea)

Public Attributes

- dng_rect **fArea**
- uint32 **fPlane**
- uint32 **fPlanes**
- int32 **fRowStep**
- int32 **fColStep**
- int32 **fPlaneStep**
- uint32 **fPixelType**
- uint32 **fPixelSize**
- uint32 **fPixelRange**
- void * **fData**
- bool **fDirty**

6.39.1 Detailed Description

Holds a buffer of pixel data with "pixel geometry" metadata.

The pixel geometry describes the layout in terms of how many planes, rows and columns plus the steps (in bytes) between each column, row and plane.

6.39.2 Member Function Documentation

6.39.2.1 `const void* dng_pixel_buffer::ConstPixel (int32 row, int32 col, uint32 plane = 0) const` [inline]

Get read-only untyped (void *) pointer to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.
col Start column for buffer pointer.
plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as void *.

6.39.2.2 `const int16* dng_pixel_buffer::ConstPixel_int16 (int32 row, int32 col, uint32 plane = 0) const` [inline]

Get read-only int16 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.
col Start column for buffer pointer.
plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as int16 *.

6.39.2.3 `const int32* dng_pixel_buffer::ConstPixel_int32 (int32 row, int32 col, uint32 plane = 0) const` [inline]

Get read-only int32 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.
col Start column for buffer pointer.
plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as int32 *.

6.39.2.4 `const int8* dng_pixel_buffer::ConstPixel_int8 (int32 row, int32 col, uint32 plane = 0) const` [inline]

Get read-only int8 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.
col Start column for buffer pointer.
plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as int8 *.

6.39.2.5 `const real32* dng_pixel_buffer::ConstPixel_real32 (int32 row, int32 col, uint32 plane = 0) const` [inline]

Get read-only real32 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.
col Start column for buffer pointer.
plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as real32 *.

6.39.2.6 `const uint16* dng_pixel_buffer::ConstPixel_uint16 (int32 row, int32 col, uint32 plane = 0) const` [inline]

Get read-only uint16 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.
col Start column for buffer pointer.
plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as uint16 *.

6.39.2.7 `const uint32* dng_pixel_buffer::ConstPixel_uint32 (int32 row, int32 col, uint32 plane = 0) const` [inline]

Get read-only uint32 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.
col Start column for buffer pointer.
plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as uint32 *.

6.39.2.8 `const uint8* dng_pixel_buffer::ConstPixel_uint8 (int32 row, int32 col, uint32 plane = 0) const` [inline]

Get read-only uint8 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.
col Start column for buffer pointer.
plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as uint8 *.

6.39.2.9 `void dng_pixel_buffer::CopyArea (const dng_pixel_buffer & src, const dng_rect & area, uint32 plane, uint32 planes)`

Copy image data from an area of one pixel buffer to same area of another.

Parameters:

src Buffer to copy from.
area Rectangle of pixel buffer to copy.
plane Plane to start copy.
planes Number of planes to copy.

6.39.2.10 `void* dng_pixel_buffer::DirtyPixel (int32 row, int32 col, uint32 plane = 0)` [inline]

Get a writable untyped (void *) pointer to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.
col Start column for buffer pointer.
plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as void *.

6.39.2.11 `int16* dng_pixel_buffer::DirtyPixel_int16 (int32 row, int32 col, uint32 plane = 0) [inline]`

Get a writable int16 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.
col Start column for buffer pointer.
plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as int16 *.

6.39.2.12 `int32* dng_pixel_buffer::DirtyPixel_int32 (int32 row, int32 col, uint32 plane = 0) [inline]`

Get a writable int32 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.
col Start column for buffer pointer.
plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as int32 *.

6.39.2.13 `int8* dng_pixel_buffer::DirtyPixel_int8 (int32 row, int32 col, uint32 plane = 0) [inline]`

Get a writable int8 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as int8 *.

6.39.2.14 `real32* dng_pixel_buffer::DirtyPixel_real32 (int32 row, int32 col, uint32 plane = 0) [inline]`

Get a writable real32 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as real32 *.

6.39.2.15 `uint16* dng_pixel_buffer::DirtyPixel_uint16 (int32 row, int32 col, uint32 plane = 0) [inline]`

Get a writable uint16 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as uint16 *.

6.39.2.16 `uint32* dng_pixel_buffer::DirtyPixel_uint32 (int32 row, int32 col, uint32 plane = 0) [inline]`

Get a writable uint32 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as uint32 *.

6.39.2.17 `uint8* dng_pixel_buffer::DirtyPixel_uint8 (int32 row, int32 col, uint32 plane = 0) [inline]`

Get a writable uint8 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.
col Start column for buffer pointer.
plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as uint8 *.

6.39.2.18 `void dng_pixel_buffer::FlipH ()`

Change metadata so pixels are iterated in opposite horizontal order. This operation does not require movement of actual pixel data.

6.39.2.19 `void dng_pixel_buffer::FlipV ()`

Change metadata so pixels are iterated in opposite vertical order. This operation does not require movement of actual pixel data.

6.39.2.20 `void dng_pixel_buffer::FlipZ ()`

Change metadata so pixels are iterated in opposite plane order. This operation does not require movement of actual pixel data.

6.39.2.21 `uint32 dng_pixel_buffer::PixelRange () const`

Get the range of pixel values.

Return values:

Range of value a pixel can take. (Meaning [0, max] for unsigned case. Signed case is biased so [-32768, max - 32768].)

6.39.2.22 `void dng_pixel_buffer::RepeatArea (const dng_rect & srcArea, const dng_rect & dstArea)`

Repeat the image data in srcArea across dstArea. (Generally used for padding operations.)

Parameters:

srcArea Area to repeat from.
dstArea Area to fill with data from srcArea.

6.39.2.23 dng_point dng_pixel_buffer::RepeatPhase (const dng_rect & srcArea, const dng_rect & dstArea) [static]

Calculate the offset phase of destination rectangle relative to source rectangle. Phase is based on a 0,0 origin and the notion of repeating srcArea across dstArea. It is the number of pixels into srcArea to start repeating from when tiling dstArea.

Return values:

dng_point containing horizontal and vertical phase.

6.39.2.24 void dng_pixel_buffer::SetZero (const dng_rect & area, uint32 plane, uint32 planes)

Initialize a rectangular area of pixel buffer to zeros.

Parameters:

area Rectangle of pixel buffer to zero.

area Area to zero

plane Plane to start filling on.

planes Number of planes to fill.

6.39.2.25 void dng_pixel_buffer::ShiftRight (uint32 shift)

Apply a right shift (C++ oerpator >>) to all pixel values. Only implemented for 16-bit (signed or unsigned) pixel buffers.

Parameters:

shift Number of bits by which to right shift each pixel value.

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_pixel_buffer.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_pixel_buffer.cpp

6.40 dng_render Class Reference

Class used to render digital negative to displayable image.

```
#include <dng_render.h>
```

Public Member Functions

- [dng_render](#) ([dng_host](#) &host, const [dng_negative](#) &negative)
- void [SetWhiteXY](#) (const [dng_xy_coord](#) &white)
- const [dng_xy_coord](#) [WhiteXY](#) () const
- void [SetExposure](#) (real64 exposure)
- real64 [Exposure](#) () const
- void [SetShadows](#) (real64 shadows)
- real64 [Shadows](#) () const
- void [SetToneCurve](#) (const [dng_1d_function](#) &curve)
- const [dng_1d_function](#) & [ToneCurve](#) () const
- void [SetFinalSpace](#) (const [dng_color_space](#) &space)
- const [dng_color_space](#) & [FinalSpace](#) () const
- void [SetFinalPixelType](#) (uint32 type)
- uint32 [FinalPixelType](#) () const
- void [SetMaximumSize](#) (uint32 size)
- uint32 [MaximumSize](#) () const
- virtual [dng_image](#) * [Render](#) ()

Protected Attributes

- [dng_host](#) & [fHost](#)
- const [dng_negative](#) & [fNegative](#)
- [dng_xy_coord](#) [fWhiteXY](#)
- real64 [fExposure](#)
- real64 [fShadows](#)
- const [dng_1d_function](#) * [fToneCurve](#)
- const [dng_color_space](#) * [fFinalSpace](#)
- uint32 [fFinalPixelType](#)
- uint32 [fMaximumSize](#)

6.40.1 Detailed Description

Class used to render digital negative to displayable image.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 [dng_render::dng_render](#) ([dng_host](#) & *host*, const [dng_negative](#) & *negative*)

Construct a rendering instance that will be used to convert a given digital negative.

Parameters:

host The host to use for memory allocation, progress updates, and abort testing.

negative The digital negative to convert to a displayable image.

6.40.3 Member Function Documentation**6.40.3.1 real64 dng_render::Exposure () const** [inline]

Get exposure compensation.

Return values:

Compensation value in stops, positive or negative.

6.40.3.2 uint32 dng_render::FinalPixelType () const [inline]

Get pixel type of final image data. Can be ttByte (default), ttShort, or ttFloat.

Return values:

Pixel type to use.

6.40.3.3 const dng_color_space& dng_render::FinalSpace () const [inline]

Get final color space in which resulting image data should be represented.

Return values:

Color space to use.

6.40.3.4 uint32 dng_render::MaximumSize () const [inline]

Get maximum dimension, in pixels, of resulting image. If the final image would have either dimension larger than this maximum, the larger of the two dimensions is set to this maximum size and the smaller dimension is adjusted to preserve the image's aspect ratio.

Return values:

Maximum allowed size.

6.40.3.5 dng_image * dng_render::Render () [virtual]

Actually render a digital negative to a displayable image. Input digital negative is passed to the constructor of this [dng_render](#) class.

Return values:

The final resulting image.

6.40.3.6 void dng_render::SetExposure (real64 *exposure*) [inline]

Set exposure compensation.

Parameters:

exposure Compensation value in stops, positive or negative.

6.40.3.7 void dng_render::SetFinalPixelType (uint32 *type*) [inline]

Set pixel type of final image data. Can be ttByte (default), ttShort, or ttFloat.

Parameters:

type Pixel type to use.

6.40.3.8 void dng_render::SetFinalSpace (const [dng_color_space](#) & *space*) [inline]

Set final color space in which resulting image data should be represented. (See [dng_color_space.h](#) for possible values.)

Parameters:

space Color space to use.

6.40.3.9 void dng_render::SetMaximumSize (uint32 *size*) [inline]

Set maximum dimension, in pixels, of resulting image. If final image would have either dimension larger than maximum, the larger of the two dimensions is set to this maximum size and the smaller dimension is adjusted to preserve aspect ratio.

Parameters:

size Maximum size to allow.

6.40.3.10 void dng_render::SetShadows (real64 *shadows*) [inline]

Set shadow clip amount.

Parameters:

shadows Shadow clip amount.

6.40.3.11 void dng_render::SetToneCurve (const [dng_1d_function](#) & *curve*) [inline]

Set custom tone curve for conversion.

Parameters:

curve 1D function that defines tone mapping to use during conversion.

6.40.3.12 void dng_renderer::SetWhiteXY (const dng_xy_coord & white)
[inline]

Set the white point to be used for conversion.

Parameters:

white White point to use.

6.40.3.13 real64 dng_renderer::Shadows () const [inline]

Get shadow clip amount.

Return values:

Shadow clip amount.

6.40.3.14 const dng_1d_function& dng_renderer::ToneCurve () const
[inline]

Get custom tone curve for conversion.

Return values:

ID function that defines tone mapping to use during conversion.

6.40.3.15 const dng_xy_coord dng_renderer::WhiteXY () const [inline]

Get the white point to be used for conversion.

Return values:

White point to use.

The documentation for this class was generated from the following files:

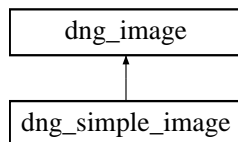
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_renderer.h
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_renderer.cpp

6.41 dng_simple_image Class Reference

[dng_image](#) derived class with simple Trim and Rotate functionality.

```
#include <dng_simple_image.h>
```

Inheritance diagram for dng_simple_image::



Public Member Functions

- **dng_simple_image** (const dng_rect &bounds, uint32 planes, uint32 pixelType, uint32 pixelRange, [dng_memory_allocator](#) &allocator)
- virtual void **Trim** (const dng_rect &r)
Trim image data outside of given bounds. Memory is not reallocated or freed.
- virtual void **Rotate** (const dng_orientation &orientation)
Rotate image according to orientation.

Protected Member Functions

- virtual void **AcquireTileBuffer** ([dng_pixel_buffer](#) &buffer, const dng_rect &tile, bool dirty) const

Protected Attributes

- [dng_pixel_buffer](#) **fBuffer**
- [AutoPtr](#)< [dng_memory_block](#) > **fMemory**

6.41.1 Detailed Description

[dng_image](#) derived class with simple Trim and Rotate functionality.

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_simple_image.h
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_simple_image.cpp

6.42 dng_sniffer_task Class Reference

Class to establish scope of a named subtask in DNG processing.

```
#include <dng_abort_sniffer.h>
```

Public Member Functions

- [dng_sniffer_task](#) ([dng_abort_sniffer](#) *sniffer, const char *name=NULL, real64 fract=0.0)
- void [Sniff](#) ()

Check for pending user cancellation or other abort. ThrowUserCanceled will be called if one is pending.

- void [UpdateProgress](#) (real64 fract)
- void [UpdateProgress](#) (uint32 done, uint32 total)
- void [Finish](#) ()

Signal task completed for progress purposes.

6.42.1 Detailed Description

Class to establish scope of a named subtask in DNG processing.

Instances of this class are intended to be stack allocated.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 `dng_sniffer_task::dng_sniffer_task (dng_abort_sniffer * sniffer, const char * name = NULL, real64 fract = 0.0) [inline]`

Inform a sniffer of a subtask in DNG processing.

Parameters:

sniffer The sniffer associated with the host on which this processing is occurring.

name The name of this subtask as a NUL terminated string.

fract Percentage of total processing this task is expected to take, from 0.0 to 1.0 .

6.42.3 Member Function Documentation

6.42.3.1 `void dng_sniffer_task::UpdateProgress (uint32 done, uint32 total) [inline]`

Update progress on this subtask.

Parameters:

done Amount of task completed in arbitrary integer units.

total Total size of task in same arbitrary integer units as done.

6.42.3.2 void dng_sniffer_task::UpdateProgress (real64 *fract*) [inline]

Update progress on this subtask.

Parameters:

fract Percentage of processing completed on current task, from 0.0 to 1.0 .

The documentation for this class was generated from the following file:

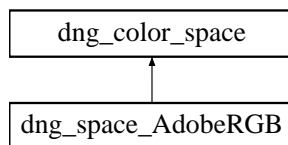
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_abort_sniffer.h](#)

6.43 dng_space_AdobeRGB Class Reference

Singleton class for AdobeRGB color space.

```
#include <dng_color_space.h>
```

Inheritance diagram for dng_space_AdobeRGB::

**Public Member Functions**

- virtual const [dng_1d_function](#) & [GammaFunction](#) () const
Returns [dng_function_GammaEncode_1_8](#).
- virtual bool [ICCPProfile](#) (uint32 &size, const uint8 *&data) const
Returns AdobeRGB (1998) ICC profile.

Static Public Member Functions

- static const [dng_color_space](#) & [Get](#) ()
Static method for getting single global instance of this color space.

6.43.1 Detailed Description

Singleton class for AdobeRGB color space.

The documentation for this class was generated from the following files:

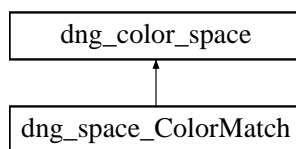
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_space.h
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_space.cpp

6.44 dng_space_ColorMatch Class Reference

Singleton class for ColorMatch color space.

```
#include <dng_color_space.h>
```

Inheritance diagram for dng_space_ColorMatch::



Public Member Functions

- virtual const [dng_1d_function](#) & [GammaFunction](#) () const
Returns [dng_function_GammaEncode_1_8](#).
- virtual bool [ICCPProfile](#) (uint32 &size, const uint8 *&data) const
Returns ColorMatch RGB ICC profile.

Static Public Member Functions

- static const [dng_color_space](#) & [Get](#) ()
Static method for getting single global instance of this color space.

6.44.1 Detailed Description

Singleton class for ColorMatch color space.

The documentation for this class was generated from the following files:

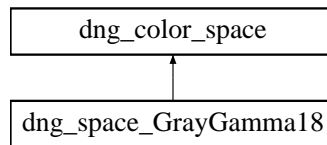
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_color_space.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_space.cpp

6.45 dng_space_GrayGamma18 Class Reference

Singleton class for gamma 1.8 grayscale color space.

```
#include <dng_color_space.h>
```

Inheritance diagram for dng_space_GrayGamma18::



Public Member Functions

- virtual const [dng_1d_function](#) & [GammaFunction](#) () const
Returns [dng_function_GammaEncode_1_8](#).
- virtual bool [ICCProfile](#) (uint32 &size, const uint8 *&data) const
Returns simple grayscale gamma 1.8 ICC profile.

Static Public Member Functions

- static const [dng_color_space](#) & [Get](#) ()
Static method for getting single global instance of this color space.

6.45.1 Detailed Description

Singleton class for gamma 1.8 grayscale color space.

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_color_space.h](#)

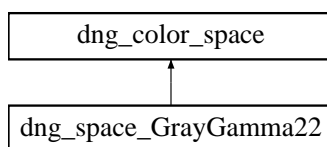
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_space.cpp

6.46 dng_space_GrayGamma22 Class Reference

Singleton class for gamma 2.2 grayscale color space.

```
#include <dng_color_space.h>
```

Inheritance diagram for dng_space_GrayGamma22::



Public Member Functions

- virtual const [dng_1d_function](#) & [GammaFunction](#) () const
Returns [dng_function_GammaEncode_2_2](#).
- virtual bool [ICCProfile](#) (uint32 &size, const uint8 *&data) const
Returns simple grayscale gamma 2.2 ICC profile.

Static Public Member Functions

- static const [dng_color_space](#) & [Get](#) ()
Static method for getting single global instance of this color space.

6.46.1 Detailed Description

Singleton class for gamma 2.2 grayscale color space.

The documentation for this class was generated from the following files:

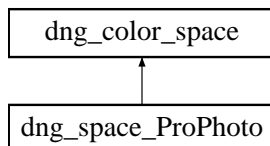
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_color_space.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_space.cpp

6.47 dng_space_ProPhoto Class Reference

Singleton class for ProPhoto RGB color space.

```
#include <dng_color_space.h>
```

Inheritance diagram for dng_space_ProPhoto::



Public Member Functions

- virtual const [dng_1d_function](#) & [GammaFunction](#) () const
Returns [dng_function_GammaEncode_1_8](#).
- virtual bool [ICCPProfile](#) (uint32 &size, const uint8 *&data) const
Returns ProPhoto RGB ICC profile.

Static Public Member Functions

- static const [dng_color_space](#) & [Get](#) ()
Static method for getting single global instance of this color space.

6.47.1 Detailed Description

Singleton class for ProPhoto RGB color space.

The documentation for this class was generated from the following files:

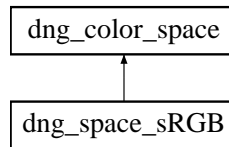
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_color_space.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_space.cpp

6.48 dng_space_sRGB Class Reference

Singleton class for sRGB color space.

```
#include <dng_color_space.h>
```

Inheritance diagram for dng_space_sRGB::



Public Member Functions

- virtual const [dng_1d_function](#) & [GammaFunction](#) () const
Returns [dng_function_GammaEncode_sRGB](#).
- virtual bool [ICCProfile](#) (uint32 &size, const uint8 *&data) const
Returns sRGB IEC61966-2.1 ICC profile.

Static Public Member Functions

- static const [dng_color_space](#) & [Get](#) ()
Static method for getting single global instance of this color space.

6.48.1 Detailed Description

Singleton class for sRGB color space.

The documentation for this class was generated from the following files:

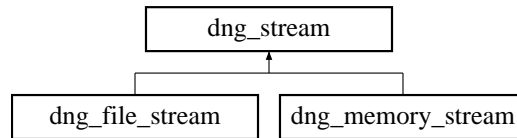
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_color_space.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_space.cpp

6.49 dng_stream Class Reference

Base stream abstraction. Has support for going between stream and pointer abstraction.

```
#include <dng_stream.h>
```

Inheritance diagram for dng_stream::



Public Types

- enum { **kInvalidOffset** = 0xFFFFFFFF }

Public Member Functions

- [dng_stream](#) (const void *data, uint32 count, uint32 offsetInOriginalFile=kInvalidOffset)
- bool [SwapBytes](#) () const
- void [SetSwapBytes](#) (bool swapBytes)
- bool [BigEndian](#) () const
- void [SetBigEndian](#) (bool bigEndian=true)
- bool [LittleEndian](#) () const
- void [SetLittleEndian](#) (bool littleEndian=true)
- uint32 [Length](#) ()
- uint32 [Position](#) () const
- uint32 [PositionInOriginalFile](#) () const
- uint32 [OffsetInOriginalFile](#) () const
- const void * [Data](#) () const

Return pointer to stream contents if the stream is entirely available as a single memory block, NULL otherwise.

- [dng_memory_block](#) * [AsMemoryBlock](#) ([dng_memory_allocator](#) &allocator)
- void [SetReadPosition](#) (uint32 offset)

Seek to a new position in stream for reading.

- void [Skip](#) (uint32 delta)
- void [Get](#) (void *data, uint32 count)
- void [SetWritePosition](#) (uint32 offset)

Seek to a new position in stream for writing.

- void [Flush](#) ()

Force any stored data in stream to be written to underlying storage.

- void [SetLength](#) (uint32 length)
- void [Put](#) (const void *data, uint32 count)

- uint8 [Get_uint8](#) ()
- void [Put_uint8](#) (uint8 x)
- uint16 [Get_uint16](#) ()
- void [Put_uint16](#) (uint16 x)
- uint32 [Get_uint32](#) ()
- void [Put_uint32](#) (uint32 x)
- int8 [Get_int8](#) ()
- void [Put_int8](#) (int8 x)
- int16 [Get_int16](#) ()
- void [Put_int16](#) (int16 x)
- int32 [Get_int32](#) ()
- void [Put_int32](#) (int32 x)
- real32 [Get_real32](#) ()
- void [Put_real32](#) (real32 x)
- real64 [Get_real64](#) ()
- void [Put_real64](#) (real64 x)
- void [Get_CString](#) (char *data, uint32 maxLength)
- void [Get_UString](#) (char *data, uint32 maxLength)
- uint32 [TagValue_uint32](#) (uint32 tagType)
- int32 [TagValue_int32](#) (uint32 tagType)
- dng_urational [TagValue_urational](#) (uint32 tagType)
- dng_srational [TagValue_srational](#) (uint32 tagType)
- real64 [TagValue_real64](#) (uint32 tagType)
- [dng_abort_sniffer](#) * [Sniffer](#) () const

Protected Types

- enum { **kDefaultBufferSize** = 4096 }

Protected Member Functions

- **dng_stream** ([dng_abort_sniffer](#) *sniffer=NULL, uint32 bufferSize=kDefault-
BufferSize, uint32 offsetInOriginalFile=kInvalidOffset)
- virtual uint32 **DoGetLength** ()
- virtual void **DoRead** (void *data, uint32 count, uint32 offset)
- virtual void **DoSetLength** (uint32 length)
- virtual void **DoWrite** (const void *data, uint32 count, uint32 offset)

6.49.1 Detailed Description

Base stream abstraction. Has support for going between stream and pointer abstraction.

Data stream abstraction for serializing and deserializing sequences of basic types and RAW image data.

6.49.2 Constructor & Destructor Documentation

6.49.2.1 dng_stream::dng_stream (const void * *data*, uint32 *count*, uint32 *offsetInOriginalFile* = kInvalidOffset)

Construct a stream with initial data.

Parameters:

data Pointer to initial contents of stream.

count Number of bytes data is valid for.

offsetInOriginalFile If data came from a file originally, offset can be saved here for later use.

6.49.3 Member Function Documentation

6.49.3.1 dng_memory_block * dng_stream::AsMemoryBlock (dng_memory_allocator & *allocator*)

Return the entire stream as a single memory block. This works for all streams, but requires copying the data to a new buffer.

Parameters:

allocator Allocator used to allocate memory.

6.49.3.2 bool dng_stream::BigEndian () const

Getter for whether data in stream is big endian.

Return values:

If true, data in stream is big endian.

6.49.3.3 void dng_stream::Get (void * *data*, uint32 *count*)

Get data from stream. Exception is thrown and no data is read if insufficient data available in stream.

Parameters:

data Buffer to put data into. Must be valid for count bytes.

count Bytes of data to read.

Exceptions:

dng_exception with `fErrorCode` equal to `dng_error_end_of_file` if not enough data in stream.

6.49.3.4 void dng_stream::Get_CString (char * *data*, uint32 *maxLength*)

Get an 8-bit character string from stream and advance read position. Routine always reads until a NUL character (8-bits of zero) is read. (That is, only *maxLength* bytes will be returned in buffer, but the stream is always advanced until a NUL is read or EOF is reached.)

Parameters:

data Buffer in which string is returned.

maxLength Maximum number of bytes to place in buffer.

Exceptions:

[*dng_exception*](#) with `fErrorCode` equal to `dng_error_end_of_file` if stream runs out before NUL is seen.

6.49.3.5 int16 dng_stream::Get_int16 () [inline]

Get one 16-bit integer from stream and advance read position. Byte swap if byte swapping is turned on.

Return values:

One 16-bit integer.

Exceptions:

[*dng_exception*](#) with `fErrorCode` equal to `dng_error_end_of_file` if not enough data in stream.

6.49.3.6 int32 dng_stream::Get_int32 () [inline]

Get one 32-bit integer from stream and advance read position. Byte swap if byte swapping is turned on.

Return values:

One 32-bit integer.

Exceptions:

[*dng_exception*](#) with `fErrorCode` equal to `dng_error_end_of_file` if not enough data in stream.

6.49.3.7 int8 dng_stream::Get_int8 () [inline]

Get one 8-bit integer from stream and advance read position. Byte swap if byte swapping is turned on.

Return values:

One 8-bit integer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

6.49.3.8 real32 dng_stream::Get_real32 ()

Get one 32-bit IEEE floating-point number from stream and advance read position. Byte swap if byte swapping is turned on.

Return values:

One 32-bit IEEE floating-point number.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

6.49.3.9 real64 dng_stream::Get_real64 ()

Get one 64-bit IEEE floating-point number from stream and advance read position. Byte swap if byte swapping is turned on.

Return values:

One 64-bit IEEE floating-point number .

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

6.49.3.10 uint16 dng_stream::Get_uint16 ()

Get an unsigned 16-bit integer from stream and advance read position. Byte swap if byte swapping is turned on.

Return values:

One unsigned 16-bit integer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

6.49.3.11 uint32 dng_stream::Get_uint32 ()

Get an unsigned 32-bit integer from stream and advance read position. Byte swap if byte swapping is turned on.

Return values:

One unsigned 16-bit integer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

6.49.3.12 uint8 dng_stream::Get_uint8 () [inline]

Get an unsigned 8-bit integer from stream and advance read position.

Return values:

One unsigned 8-bit integer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

6.49.3.13 void dng_stream::Get_UString (char * *data*, uint32 *maxLength*)

Get a 16-bit character string from stream and advance read position. 16-bit characters are truncated to 8-bits. Routine always reads until a NUL character (16-bits of zero) is read. (That is, only *maxLength* bytes will be returned in buffer, but the stream is always advanced until a NUL is read or EOF is reached.)

Parameters:

data Buffer to place string in.

maxLength Maximum number of bytes to place in buffer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if stream runs out before NUL is seen.

6.49.3.14 uint32 dng_stream::Length ()

Getter for length of data in stream.

Return values:

Length of readable data in stream.

6.49.3.15 bool dng_stream::LittleEndian () const [inline]

Getter for whether data in stream is big endian.

Return values:

If true, data in stream is big endian.

6.49.3.16 uint32 dng_stream::OffsetInOriginalFile () const

Getter for offset in original file.

Return values:

kInvalidOffset if no offset in original file is set, offset in original file otherwise.

6.49.3.17 uint32 dng_stream::Position () const [inline]

Getter for current offset in stream.

Return values:

current offset from start of stream.

6.49.3.18 uint32 dng_stream::PositionInOriginalFile () const

Getter for current position in original file, taking into account OffsetInOriginalFile stream data was taken from.

Return values:

kInvalidOffset if no offset in original file is set, sum of offset in original file and current position otherwise.

6.49.3.19 void dng_stream::Put (const void * data, uint32 count)

Write data to stream.

Parameters:

data Buffer of data to write to stream.

count Bytes of in data.

6.49.3.20 void dng_stream::Put_int16 (int16 x) [inline]

Put one 16-bit integer to stream and advance write position. Byte swap if byte swapping is turned on.

Parameters:

x One 16-bit integer.

6.49.3.21 void dng_stream::Put_int32 (int32 *x*) [inline]

Put one 32-bit integer to stream and advance write position. Byte swap if byte swapping is turned on.

Parameters:

x One 32-bit integer.

6.49.3.22 void dng_stream::Put_int8 (int8 *x*) [inline]

Put one 8-bit integer to stream and advance write position. Byte swap if byte swapping is turned on.

Parameters:

x One 8-bit integer.

6.49.3.23 void dng_stream::Put_real32 (real32 *x*)

Put one 32-bit IEEE floating-point number to stream and advance write position. Byte swap if byte swapping is turned on.

Parameters:

x One 32-bit IEEE floating-point number.

6.49.3.24 void dng_stream::Put_real64 (real64 *x*)

Put one 64-bit IEEE floating-point number to stream and advance write position. Byte swap if byte swapping is turned on.

Parameters:

x One 64-bit IEEE floating-point number.

6.49.3.25 void dng_stream::Put_uint16 (uint16 *x*)

Put an unsigned 16-bit integer to stream and advance write position. Byte swap if byte swapping is turned on.

Parameters:

x One unsigned 16-bit integer.

6.49.3.26 void dng_stream::Put_uint32 (uint32 *x*)

Put an unsigned 32-bit integer to stream and advance write position. Byte swap if byte swapping is turned on.

Parameters:

x One unsigned 16-bit integer.

6.49.3.27 void dng_stream::Put_uint8 (uint8 *x*) [inline]

Put an unsigned 8-bit integer to stream and advance write position.

Parameters:

x One unsigned 8-bit integer.

6.49.3.28 void dng_stream::SetBigEndian (bool *bigEndian* = true)

Setter for whether data in stream is big endian.

Parameters:

bigEndian If true, data in stream is big endian.

6.49.3.29 void dng_stream::SetLength (uint32 *length*)

Set length of available data.

Parameters:

length Number of bytes of available data in stream.

6.49.3.30 void dng_stream::SetLittleEndian (bool *littleEndian* = true) [inline]

Setter for whether data in stream is big endian.

Parameters:

littleEndian If true, data in stream is big endian.

6.49.3.31 void dng_stream::SetSwapBytes (bool *swapBytes*) [inline]

Setter for whether stream is swapping byte order on input/output.

Parameters:

swapBytes If true, stream will swap byte order on input or output for future reads/writes.

6.49.3.32 void dng_stream::Skip (uint32 *delta*) [inline]

Skip forward in stream.

Parameters:

delta Number of bytes to skip forward.

6.49.3.33 dng_abort_sniffer* dng_stream::Sniffer () const [inline]

Getter for sniffer associated with stream.

Return values:

The sniffer for this stream.

6.49.3.34 bool dng_stream::SwapBytes () const [inline]

Getter for whether stream is swapping byte order on input/output.

Return values:

If true, data will be swapped on input/output.

6.49.3.35 int32 dng_stream::TagValue_int32 (uint32 *tagType*)

Get a value of size indicated by tag type from stream and advance read position. Byte swap if byte swapping is turned on and tag type is larger than a byte. Value is returned as a 32-bit integer.

Parameters:

tagType Tag type of data stored in stream.

Return values:

One 32-bit integer.

Exceptions:

dng_exception with `fErrorCode` equal to `dng_error_end_of_file` if not enough data in stream.

6.49.3.36 real64 dng_stream::TagValue_real64 (uint32 *tagType*)

Get a value of size indicated by tag type from stream and advance read position. Byte swap if byte swapping is turned on and tag type is larger than a byte. Value is returned as a 64-bit IEEE floating-point number.

Parameters:

tagType Tag type of data stored in stream.

Return values:

One 64-bit IEEE floating-point number.

Exceptions:

dng_exception with `fErrorCode` equal to `dng_error_end_of_file` if not enough data in stream.

6.49.3.37 dng_srational dng_stream::TagValue_srational (uint32 tagType)

Get a value of size indicated by tag type from stream and advance read position. Byte swap if byte swapping is turned on and tag type is larger than a byte. Value is returned as a `dng_srational`.

Parameters:

tagType Tag type of data stored in stream.

Return values:

One `dng_srational`.

Exceptions:

dng_exception with `fErrorCode` equal to `dng_error_end_of_file` if not enough data in stream.

6.49.3.38 uint32 dng_stream::TagValue_uint32 (uint32 tagType)

Get a value of size indicated by tag type from stream and advance read position. Byte swap if byte swapping is turned on and tag type is larger than a byte. Value is returned as an unsigned 32-bit integer.

Parameters:

tagType Tag type of data stored in stream.

Return values:

One unsigned 32-bit integer.

Exceptions:

dng_exception with `fErrorCode` equal to `dng_error_end_of_file` if not enough data in stream.

6.49.3.39 dng_urational dng_stream::TagValue_urational (uint32 tagType)

Get a value of size indicated by tag type from stream and advance read position. Byte swap if byte swapping is turned on and tag type is larger than a byte. Value is returned as a dng_urational.

Parameters:

tagType Tag type of data stored in stream.

Return values:

One dng_urational.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

The documentation for this class was generated from the following files:

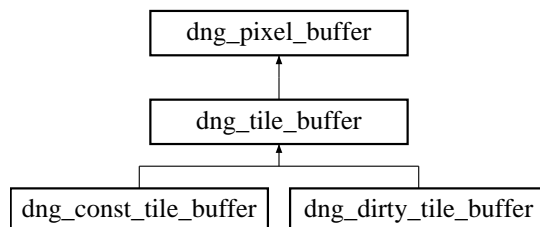
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_stream.h
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_stream.cpp

6.50 dng_tile_buffer Class Reference

Class to get resource acquisition is instantiation behavior for tile buffers. Can be dirty or constant tile access.

```
#include <dng_image.h>
```

Inheritance diagram for dng_tile_buffer::

**Protected Member Functions**

- *dng_tile_buffer* (const *dng_image* &image, const dng_rect &tile, bool dirty)

Protected Attributes

- const *dng_image* & fImage

6.50.1 Detailed Description

Class to get resource acquisition is instantiation behavior for tile buffers. Can be dirty or constant tile access.

6.50.2 Constructor & Destructor Documentation

6.50.2.1 `dng_tile_buffer::dng_tile_buffer (const dng_image & image, const dng_rect & tile, bool dirty)` `[protected]`

Obtain a tile from an image.

Parameters:

image Image tile will come from.

tile Rectangle denoting extent of tile.

dirty Flag indicating whether this is read-only or read-write access.

The documentation for this class was generated from the following files:

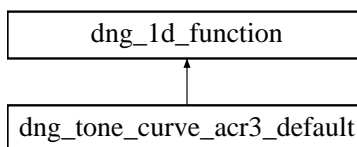
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_image.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_image.cpp

6.51 dng_tone_curve_acr3_default Class Reference

Default ACR3 tone curve.

```
#include <dng_render.h>
```

Inheritance diagram for `dng_tone_curve_acr3_default`:



Public Member Functions

- virtual real64 [Evaluate](#) (real64 x) const
Returns output value for a given input tone.
- virtual real64 [EvaluateInverse](#) (real64 x) const
Returns nearest input value for a given output tone.

Static Public Member Functions

- static const [dng_1d_function](#) & Get ()

6.51.1 Detailed Description

Default ACR3 tone curve.

The documentation for this class was generated from the following files:

- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/[dng_render.h](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_render.cpp

7 DNG SDK File Documentation

7.1 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_1d_function.h File Reference

Classes

- class [dng_1d_function](#)
A 1D floating-point function.
- class [dng_1d_identity](#)
An identity ($x \rightarrow y$ such that $x == y$ for all x) mapping function.
- class [dng_1d_concatenate](#)
A [dng_1d_function](#) that represents the composition (curry) of two other [dng_1d_functions](#).

7.1.1 Detailed Description

Classes for a 1D floating-point to floating-point function abstraction.

7.2 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_1d_table.h File Reference

Classes

- class [dng_1d_table](#)

A 1D floating-point lookup table using linear interpolation.

7.2.1 Detailed Description

Definition of a lookup table based 1D floating-point to floating-point function abstraction using linear interpolation.

7.3 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_abort_sniffer.h File Reference

Classes

- class [dng_abort_sniffer](#)
Class for signaling user cancellation and receiving progress updates.
- class [dng_sniffer_task](#)
Class to establish scope of a named subtask in DNG processing.

7.3.1 Detailed Description

Classes supporting user cancellation and progress tracking.

7.4 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_area_task.h File Reference

Classes

- class [dng_area_task](#)
Abstract class for rectangular processing operations with support for partitioning across multiple processing resources and observing memory constraints.

7.4.1 Detailed Description

Class to handle partitioning a rectangular image processing operation taking into account multiple processing resources and memory constraints.

7.5 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_assertions.h File Reference

Defines

- #define [ASSERT](#)(x, y)
- #define [REPORT](#)(x) [ASSERT](#) (false, x)

7.5.1 Detailed Description

Conditionally compiled assertion check support.

7.5.2 Define Documentation

7.5.2.1 #define [ASSERT](#)(x, y)

Conditionally compiled macro to check an assertion and display a message if it fails and assertions are compiled in via qDNGDebug

Parameters:

- x* Predicate which must be true.
- y* String to display if x is not true.

7.6 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_auto_ptr.h File Reference

Classes

- class [AutoPtr](#)< T >
A class intended to be used in stack scope to hold a pointer from new. The held pointer will be deleted automatically if the scope is left without calling Release on the [AutoPtr](#) first.

7.6.1 Detailed Description

Class to implement std::auto_ptr like functionality even on platforms which do not have a full Standard C++ library.

7.7 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_bottlenecks.h File Reference**Classes**

- struct **dng_suite**

Typedefs

- typedef void() **ZeroBytesProc** (void *dPtr, uint32 count)
- typedef void() **CopyBytesProc** (const void *sPtr, void *dPtr, uint32 count)
- typedef void() **SwapBytes16Proc** (uint16 *dPtr, uint32 count)
- typedef void() **SwapBytes32Proc** (uint32 *dPtr, uint32 count)
- typedef void() **SetArea8Proc** (uint8 *dPtr, uint8 value, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep)
- typedef void() **SetArea16Proc** (uint16 *dPtr, uint16 value, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep)
- typedef void() **SetArea32Proc** (uint32 *dPtr, uint32 value, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep)
- typedef void() **CopyArea8Proc** (const uint8 *sPtr, uint8 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- typedef void() **CopyArea16Proc** (const uint16 *sPtr, uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- typedef void() **CopyArea32Proc** (const uint32 *sPtr, uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- typedef void() **CopyArea8_16Proc** (const uint8 *sPtr, uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- typedef void() **CopyArea8_S16Proc** (const uint8 *sPtr, int16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- typedef void() **CopyArea8_32Proc** (const uint8 *sPtr, uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- typedef void() **CopyArea16_S16Proc** (const uint16 *sPtr, int16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- typedef void() **CopyArea16_32Proc** (const uint16 *sPtr, uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)

- typedef void() **CopyArea8_R32Proc** (const uint8 *sPtr, real32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- typedef void() **CopyArea16_R32Proc** (const uint16 *sPtr, real32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- typedef void() **CopyAreaS16_R32Proc** (const int16 *sPtr, real32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- typedef void() **CopyAreaR32_8Proc** (const real32 *sPtr, uint8 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- typedef void() **CopyAreaR32_16Proc** (const real32 *sPtr, uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- typedef void() **CopyAreaR32_S16Proc** (const real32 *sPtr, int16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- typedef void() **RepeatArea8Proc** (const uint8 *sPtr, uint8 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep, uint32 repeatV, uint32 repeatH, uint32 phaseV, uint32 phaseH)
- typedef void() **RepeatArea16Proc** (const uint16 *sPtr, uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep, uint32 repeatV, uint32 repeatH, uint32 phaseV, uint32 phaseH)
- typedef void() **RepeatArea32Proc** (const uint32 *sPtr, uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep, uint32 repeatV, uint32 repeatH, uint32 phaseV, uint32 phaseH)
- typedef void() **ShiftRight16Proc** (uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep, uint32 shift)
- typedef void() **BilinearRow16Proc** (const uint16 *sPtr, uint16 *dPtr, uint32 cols, uint32 patPhase, uint32 patCount, const uint32 *kernCounts, const int32 *const *kernOffsets, const uint16 *const *kernWeights, uint32 sShift)
- typedef void() **BilinearRow32Proc** (const real32 *sPtr, real32 *dPtr, uint32 cols, uint32 patPhase, uint32 patCount, const uint32 *kernCounts, const int32 *const *kernOffsets, const real32 *const *kernWeights, uint32 sShift)
- typedef void() **BaselineABCtoRGBProc** (const real32 *sPtrA, const real32 *sPtrB, const real32 *sPtrC, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const dng_vector &cameraWhite, const dng_matrix &cameraToRGB)
- typedef void() **BaselineABCDtoRGBProc** (const real32 *sPtrA, const real32 *sPtrB, const real32 *sPtrC, const real32 *sPtrD, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const dng_vector &cameraWhite, const dng_matrix &cameraToRGB)
- typedef void() **BaselineGrayToRGBProc** (const real32 *sPtrR, const real32 *sPtrG, const real32 *sPtrB, real32 *dPtrG, uint32 count, const dng_matrix &matrix)

7.7

C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_bottlenecks.h
File Reference 144

- typedef void() **BaselineRGBtoRGBProc** (const real32 *sPtrR, const real32 *sPtrG, const real32 *sPtrB, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const dng_matrix &matrix)
- typedef void() **Baseline1DTableProc** (const real32 *sPtr, real32 *dPtr, uint32 count, const [dng_1d_table](#) &table)
- typedef void() **BaselineRGBToneProc** (const real32 *sPtrR, const real32 *sPtrG, const real32 *sPtrB, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const [dng_1d_table](#) &table)
- typedef void() **ResampleDown16Proc** (const uint16 *sPtr, uint16 *dPtr, uint32 sCount, int32 sRowStep, const int16 *wPtr, uint32 wCount, uint32 pixelRange)
- typedef void() **ResampleDown32Proc** (const real32 *sPtr, real32 *dPtr, uint32 sCount, int32 sRowStep, const real32 *wPtr, uint32 wCount)
- typedef void() **ResampleAcross16Proc** (const uint16 *sPtr, uint16 *dPtr, uint32 dCount, const int32 *coord, const int16 *wPtr, uint32 wCount, uint32 wStep, uint32 pixelRange)
- typedef void() **ResampleAcross32Proc** (const real32 *sPtr, real32 *dPtr, uint32 dCount, const int32 *coord, const real32 *wPtr, uint32 wCount, uint32 wStep)

Functions

- void **DoZeroBytes** (void *dPtr, uint32 count)
- void **DoCopyBytes** (const void *sPtr, void *dPtr, uint32 count)
- void **DoSwapBytes16** (uint16 *dPtr, uint32 count)
- void **DoSwapBytes32** (uint32 *dPtr, uint32 count)
- void **DoSetArea8** (uint8 *dPtr, uint8 value, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep)
- void **DoSetArea16** (uint16 *dPtr, uint16 value, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep)
- void **DoSetArea32** (uint32 *dPtr, uint32 value, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep)
- void **DoCopyArea8** (const uint8 *sPtr, uint8 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- void **DoCopyArea16** (const uint16 *sPtr, uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- void **DoCopyArea32** (const uint32 *sPtr, uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- void **DoCopyArea8_16** (const uint8 *sPtr, uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)

- void **DoCopyArea8_S16** (const uint8 *sPtr, int16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- void **DoCopyArea8_32** (const uint8 *sPtr, uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- void **DoCopyArea16_S16** (const uint16 *sPtr, int16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- void **DoCopyArea16_32** (const uint16 *sPtr, uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- void **DoCopyArea8_R32** (const uint8 *sPtr, real32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- void **DoCopyArea16_R32** (const uint16 *sPtr, real32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- void **DoCopyAreaS16_R32** (const int16 *sPtr, real32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- void **DoCopyAreaR32_8** (const real32 *sPtr, uint8 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- void **DoCopyAreaR32_16** (const real32 *sPtr, uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- void **DoCopyAreaR32_S16** (const real32 *sPtr, int16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- void **DoRepeatArea8** (const uint8 *sPtr, uint8 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep, uint32 repeatV, uint32 repeatH, uint32 phaseV, uint32 phaseH)
- void **DoRepeatArea16** (const uint16 *sPtr, uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep, uint32 repeatV, uint32 repeatH, uint32 phaseV, uint32 phaseH)
- void **DoRepeatArea32** (const uint32 *sPtr, uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep, uint32 repeatV, uint32 repeatH, uint32 phaseV, uint32 phaseH)
- void **DoShiftRight16** (uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep, uint32 shift)
- void **DoBilinearRow16** (const uint16 *sPtr, uint16 *dPtr, uint32 cols, uint32 patPhase, uint32 patCount, const uint32 *kernCounts, const int32 *const *kernOffsets, const uint16 *const *kernWeights, uint32 sShift)

7.7

- void **DoBilinearRow32** (const real32 *sPtr, real32 *dPtr, uint32 cols, uint32 patPhase, uint32 patCount, const uint32 *kernCounts, const int32 *const *kernOffsets, const real32 *const *kernWeights, uint32 sShift)
- void **DoBaselineABCToRGB** (const real32 *sPtrA, const real32 *sPtrB, const real32 *sPtrC, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const dng_vector &cameraWhite, const dng_matrix &cameraToRGB)
- void **DoBaselineABCDtoRGB** (const real32 *sPtrA, const real32 *sPtrB, const real32 *sPtrC, const real32 *sPtrD, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const dng_vector &cameraWhite, const dng_matrix &cameraToRGB)
- void **DoBaselineRGBtoGray** (const real32 *sPtrR, const real32 *sPtrG, const real32 *sPtrB, real32 *dPtrG, uint32 count, const dng_matrix &matrix)
- void **DoBaselineRGBtoRGB** (const real32 *sPtrR, const real32 *sPtrG, const real32 *sPtrB, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const dng_matrix &matrix)
- void **DoBaseline1DTable** (const real32 *sPtr, real32 *dPtr, uint32 count, const [dng_1d_table](#) &table)
- void **DoBaselineRGBTone** (const real32 *sPtrR, const real32 *sPtrG, const real32 *sPtrB, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const [dng_1d_table](#) &table)
- void **DoResampleDown16** (const uint16 *sPtr, uint16 *dPtr, uint32 sCount, int32 sRowStep, const int16 *wPtr, uint32 wCount, uint32 pixelRange)
- void **DoResampleDown32** (const real32 *sPtr, real32 *dPtr, uint32 sCount, int32 sRowStep, const real32 *wPtr, uint32 wCount)
- void **DoResampleAcross16** (const uint16 *sPtr, uint16 *dPtr, uint32 dCount, const int32 *coord, const int16 *wPtr, uint32 wCount, uint32 wStep, uint32 pixelRange)
- void **DoResampleAcross32** (const real32 *sPtr, real32 *dPtr, uint32 dCount, const int32 *coord, const real32 *wPtr, uint32 wCount, uint32 wStep)

Variables

- dng_suite **gDNGSuite**

7.7.1 Detailed Description

Indirection mechanism for performance-critical routines that might be replaced with hand-optimized or hardware-specific implementations.

7.8 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_camera_profile.h File Reference

Classes

- class [dng_camera_profile](#)
Container for DNG camera color profile and calibration data.

Variables

- const char * **kProfileName_Embedded**

7.8.1 Detailed Description

Support for DNG camera color profile information. Per the [DNG 1.1.0 specification](#), a DNG file can store up to two sets of color profile information for a camera in the DNG file from that camera. The second set is optional and when there are two sets, they represent profiles made under different illumination.

Profiling information is optionally separated into two parts. One part represents a profile for a reference camera. (ColorMatrix1 and ColorMatrix2 here.) The second is a per-camera calibration that takes into account unit-to-unit variation. This is designed to allow replacing the reference color matrix with one of one's own construction while maintaining any unit-specific calibration the camera manufacturer may have provided.

See Appendix 6 of the [DNG 1.1.0 specification](#) for more information.

7.9 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_space.h File Reference

Classes

- class [dng_function_GammaEncode_sRGB](#)
A [dng_1d_function](#) for gamma encoding in sRGB color space.
- class [dng_function_GammaEncode_1_8](#)
A [dng_1d_function](#) for gamma encoding with 1.8 gamma.
- class [dng_function_GammaEncode_2_2](#)
A [dng_1d_function](#) for gamma encoding with 2.2 gamma.
- class [dng_color_space](#)

7.10

C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_spec.h File Reference

148

An abstract color space.

- class [dng_space_sRGB](#)
Singleton class for sRGB color space.
- class [dng_space_AdobeRGB](#)
Singleton class for AdobeRGB color space.
- class [dng_space_ColorMatch](#)
Singleton class for ColorMatch color space.
- class [dng_space_ProPhoto](#)
Singleton class for ProPhoto RGB color space.
- class [dng_space_GrayGamma18](#)
Singleton class for gamma 1.8 grayscale color space.
- class [dng_space_GrayGamma22](#)
Singleton class for gamma 2.2 grayscale color space.

7.9.1 Detailed Description

Standard gamma functions and color spaces used within the DNG SDK.

7.10 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_spec.h File Reference

Classes

- class [dng_color_spec](#)
Color transform taking into account white point and camera calibration and individual calibration from DNG negative.

Functions

- [dng_matrix_3by3](#) [MapWhiteMatrix](#) (const dng_xy_coord &white1, const dng_xy_coord &white2)
Compute a 3x3 matrix which maps colors from white point white1 to white point white2.

7.11

C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_date_time.h
File Reference 149

7.10.1 Detailed Description

Class for holding a specific color transform.

7.10.2 Function Documentation

7.10.2.1 dng_matrix_3by3 MapWhiteMatrix (const dng_xy_coord & white1, const dng_xy_coord & white2)

Compute a 3x3 matrix which maps colors from white point white1 to white point white2.

Uses linearized Bradford adaptation matrix to compute a mapping from colors measured with one white point (white1) to another (white2).

7.11 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_date_time.h File Reference

Classes

- class [dng_date_time](#)
Class for holding a date/time and converting to and from relevant date/time formats.
- class [dng_date_time_storage_info](#)
Store file offset from which date was read.

Enumerations

- enum [dng_date_time_format](#) { [dng_date_time_format_unknown](#) = 0, [dng_date_time_format_exif](#) = 1, [dng_date_time_format_unix_little_endian](#) = 2, [dng_date_time_format_unix_big_endian](#) = 3 }
- Tag to encode date representation format.*

Functions

- void [CurrentDateTimeAndZone](#) (dng_date_time &dt, int32 &zone)
- void [DecodeUnixTime](#) (uint32 unixTime, dng_date_time &dt)
Convert UNIX "seconds since Jan 1, 1970" time to a [dng_date_time](#).
- int32 [LocalTimeZone](#) (const dng_date_time &dt)

7.11.1 Detailed Description

Functions and classes for working with dates and times in DNG files.

7.11.2 Enumeration Type Documentation

7.11.2.1 enum [dng_date_time_format](#)

Tag to encode date representation format.

Enumerator:

dng_date_time_format_exif Date format not known.

dng_date_time_format_unix_little_endian EXIF date string.

dng_date_time_format_unix_big_endian 32-bit UNIX time as 4-byte little endian

7.11.3 Function Documentation

7.11.3.1 void [CurrentDateTimeAndZone](#) ([dng_date_time](#) & *dt*, int32 & *zone*)

Get the current date/time and timezone.

Parameters:

dt Receives current data/time

zone Seconds from GMT of current timezone.

7.11.3.2 int32 [LocalTimeZone](#) (const [dng_date_time](#) & *dt*)

Return timezone of current location at a given date.

Parameters:

dt Date at which to compute timezone difference. (For example, used to determine Daylight Savings, etc.)

Return values:

Number of seconds difference between local time zone and GMT at date/time *dt*.

7.12 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_errors.h File Reference

Typedefs

- typedef int32 [dng_error_code](#)

Type for all errors used in DNG SDK. Generally held inside a [dng_exception](#).

Enumerations

- enum {
 dng_error_none = 0, dng_error_unknown = 100000, dng_error_not_yet_implemented, dng_error_silent,
 dng_error_user_canceled, dng_error_host_insufficient, dng_error_memory, dng_error_bad_format,
 dng_error_matrix_math, dng_error_open_file, dng_error_read_file, dng_error_write_file,
 dng_error_end_of_file }

7.12.1 Detailed Description

Error code values.

7.13 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_exceptions.h File Reference**Classes**

- class [dng_exception](#)
 All exceptions thrown by the DNG SDK use this exception class.

Functions

- void [Throw_dng_error](#) (dng_error_code err)
 Throw an exception based on an arbitrary error code.
- void [Fail_dng_error](#) (dng_error_code err)
 Convenience function to throw [dng_exception](#) with error code if error_code is not dng_error_none .
- void [ThrowProgramError](#) ()
 Convenience function to throw [dng_exception](#) with error code dng_error_unknown .
- void [ThrowNotYetImplemented](#) ()
 Convenience function to throw [dng_exception](#) with error code dng_error_not_yet_implemented .
- void [ThrowSilentError](#) ()

Convenience function to throw [dng_exception](#) with error code `dng_error_silent`.

- void [ThrowUserCanceled](#) ()

Convenience function to throw [dng_exception](#) with error code `dng_error_user_canceled`.

- void [ThrowHostInsufficient](#) ()

Convenience function to throw [dng_exception](#) with error code `dng_error_host_insufficient`.

- void [ThrowMemoryFull](#) ()

Convenience function to throw [dng_exception](#) with error code `dng_error_memory`.

- void [ThrowBadFormat](#) ()

Convenience function to throw [dng_exception](#) with error code `dng_error_bad_format`.

- void [ThrowMatrixMath](#) ()

Convenience function to throw [dng_exception](#) with error code `dng_error_matrix_math`.

- void [ThrowOpenFile](#) ()

Convenience function to throw [dng_exception](#) with error code `dng_error_open_file`.

- void [ThrowReadFile](#) ()

Convenience function to throw [dng_exception](#) with error code `dng_error_read_file`.

- void [ThrowWriteFile](#) ()

Convenience function to throw [dng_exception](#) with error code `dng_error_write_file`.

- void [ThrowEndOfFile](#) ()

Convenience function to throw [dng_exception](#) with error code `dng_error_end_of_file`.

7.13.1 Detailed Description

C++ exception support for DNG SDK.

7.14 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_exif.h File Reference

Classes

- class [dng_exif](#)
Container class for parsing and holding EXIF tags.

7.14.1 Detailed Description

EXIF read access support. See the [EXIF specification](#) for full description of tags.

7.15 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_fast_module.h File Reference

7.15.1 Detailed Description

Include file to set optimization to highest level for performance-critical routines. Normal files should have optimization set to normal level to save code size as there is less cache pollution this way.

7.16 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_file_stream.h File Reference

Classes

- class [dng_file_stream](#)
A stream to/from a disk file. See [dng_stream](#) for read/write interface.

7.16.1 Detailed Description

Simple, portable, file read/write support.

7.17 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_filter_task.h File Reference

Classes

- class [dng_filter_task](#)

7.18

C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_fingerprint.h File Reference 154

Represents a task which filters an area of a source [dng_image](#) to an area of a destination [dng_image](#).

7.17.1 Detailed Description

Specialization of [dng_area_task](#) for processing an area from one [dng_image](#) to an area of another.

7.18 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_fingerprint.h File Reference

Classes

- class [dng_fingerprint](#)
Container fingerprint (MD5 only at present).
- class **MD5Fingerprint**

7.18.1 Detailed Description

Fingerprint (cryptographic hashing) support for generating strong hashes of image data.

7.19 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_flags.h File Reference

Defines

- #define **qDNGDebug** 0
- #define **qDNGLittleEndian** !qDNGBigEndian
- #define **qDNGThreadSafe** (qMacOS || qWinOS)
- #define **qDNGValidateTarget** 0
- #define **qDNGValidate** qDNGValidateTarget

7.19.1 Detailed Description

Conditional compilation flags for DNG SDK.

All conditional compilation macros for the DNG SDK begin with a lowercase 'q'.

7.20

C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_globals.h
File Reference

155

7.20 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_globals.h File Reference

7.20.1 Detailed Description

Definitions of global variables controlling DNG SDK behavior. Currently only used for validation control.

7.21 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_host.h File Reference

Classes

- class [dng_host](#)

The main class for communication between the application and the DNG SDK. Used to customize memory allocation and other behaviors.

7.21.1 Detailed Description

Class definition for [dng_host](#), initial point of contact and control between host application and DNG SDK.

7.22 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_ifd.h File Reference

Classes

- class [dng_ifd](#)

Container for a single image file directory of a digital negative.

7.22.1 Detailed Description

DNG image file directory support.

7.23 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_image.h File Reference

Classes

- class [dng_tile_buffer](#)
Class to get resource acquisition is instantiation behavior for tile buffers. Can be dirty or constant tile access.
- class [dng_const_tile_buffer](#)
Class to get resource acquisition is instantiation behavior for constant (read-only) tile buffers.
- class [dng_dirty_tile_buffer](#)
Class to get resource acquisition is instantiation behavior for dirty (writable) tile buffers.
- class [dng_image](#)
Base class for holding image data in DNG SDK. See [dng_simple_image](#) for derived class most often used in DNG SDK.

7.23.1 Detailed Description

Support for working with image data in DNG SDK.

7.24 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_image_writer.h File Reference

Classes

- class **dng_jpeg_preview**
- class **dng_resolution**
- class **tiff_tag**
- class **tag_data_ptr**
- class **tag_string**
- class **tag_encoded_text**
- class **tag_uint8**
- class **tag_uint8_ptr**
- class **tag_uint16**
- class **tag_int16_ptr**
- class **tag_uint16_ptr**
- class **tag_uint32**

- class **tag_uint32_ptr**
- class **tag_urational**
- class **tag_urational_ptr**
- class **tag_srational**
- class **tag_srational_ptr**
- class **tag_matrix**
- class **tag_icc_profile**
- class **tag_cfa_pattern**
- class **tag_exif_date_time**
- class **tag_iptc**
- class **tag_xmp**
- class **tag_adobe_data**
- class **tiff_directory**
- class **basic_tag_set**
- class **exif_tag_set**
- class [dng_image_writer](#)

Support for writing [dng_image](#) or [dng_negative](#) instances to a [dng_stream](#) in TIFF or DNG format.

7.24.1 Detailed Description

Support for writing DNG images to files.

7.25 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_info.h File Reference

Classes

- class [dng_info](#)

Top-level structure of DNG file with access to metadata.

7.25.1 Detailed Description

Class for holding top-level information about a DNG image.

7.26 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_iptc.h File Reference

Classes

- class [dng_iptc](#)
Class for reading and holding IPTC metadata associated with a DNG file.

7.26.1 Detailed Description

Support for IPTC metadata within DNG files.

7.27 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_linearization_info.h File Reference

Classes

- class [dng_linearization_info](#)
Class for managing data values related to DNG linearization.

7.27.1 Detailed Description

Support for linearization table and black level tags.

7.28 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_matrix.h File Reference

Classes

- class **dng_matrix**
- class **dng_matrix_3by3**
- class **dng_matrix_4by3**
- class **dng_vector**
- class **dng_vector_3**

Functions

- **dng_matrix operator *** (const dng_matrix &A, const dng_matrix &B)
- **dng_vector operator *** (const dng_matrix &A, const dng_vector &B)

7.29 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_memory_stream.h File

Reference

159

- dng_matrix **operator** * (real64 scale, const dng_matrix &A)
- dng_vector **operator** * (real64 scale, const dng_vector &A)
- dng_matrix **operator**+ (const dng_matrix &A, const dng_matrix &B)
- dng_matrix **Transpose** (const dng_matrix &A)
- dng_matrix **Invert** (const dng_matrix &A)
- dng_matrix **Invert** (const dng_matrix &A, const dng_matrix &hint)
- real64 **MaxEntry** (const dng_matrix &A)
- real64 **MaxEntry** (const dng_vector &A)
- real64 **MinEntry** (const dng_matrix &A)
- real64 **MinEntry** (const dng_vector &A)

7.28.1 Detailed Description

Matrix and vector classes, including specialized 3x3 and 4x3 versions as well as length 3 vectors.

7.29 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_memory_stream.h File Reference

Classes

- class [dng_memory_stream](#)
A [dng_stream](#) which can be read from or written to memory.

7.29.1 Detailed Description

Stream abstraction to/from in-memory data.

7.30 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_mosaic_info.h File Reference

Classes

- class [dng_mosaic_info](#)
Support for describing color filter array patterns and manipulating mosaic sample data.

7.30.1 Detailed Description

Support for descriptive information about color filter array patterns.

7.31 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_negative.h File Reference

Classes

- class [dng_negative](#)

Main class for holding DNG image data and associated metadata.

7.31.1 Detailed Description

7.32 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_pixel_buffer.h File Reference

Classes

- class [dng_pixel_buffer](#)

Holds a buffer of pixel data with "pixel geometry" metadata.

Functions

- void [OptimizeOrder](#) (const void *&sPtr, void *&dPtr, uint32 sPixelSize, uint32 dPixelSize, uint32 &count0, uint32 &count1, uint32 &count2, int32 &sStep0, int32 &sStep1, int32 &sStep2, int32 &dStep0, int32 &dStep1, int32 &dStep2)
Compute best set of step values for a given source and destination area and stride.
- void **OptimizeOrder** (const void *&sPtr, uint32 sPixelSize, uint32 &count0, uint32 &count1, uint32 &count2, int32 &sStep0, int32 &sStep1, int32 &sStep2)
- void **OptimizeOrder** (void *&dPtr, uint32 dPixelSize, uint32 &count0, uint32 &count1, uint32 &count2, int32 &dStep0, int32 &dStep1, int32 &dStep2)

7.32.1 Detailed Description

Support for holding buffers of sample data.

7.33 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_read_image.h File Reference

Classes

- class [dng_read_image](#)

7.33.1 Detailed Description

Support for DNG image reading.

7.34 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_render.h File Reference

Classes

- class [dng_function_exposure_ramp](#)
Curve for pre-exposure-compensation adjustment based on noise floor, shadows, and highlight level.
- class [dng_function_exposure_tone](#)
Exposure compensation curve for a given compensation amount in stops using quadric for roll-off.
- class [dng_tone_curve_acr3_default](#)
Default ACR3 tone curve.
- class [dng_function_gamma_encode](#)
Encoding gamma curve for a given color space.
- class [dng_render](#)
Class used to render digital negative to displayable image.

7.34.1 Detailed Description

Classes for conversion of RAW data to final image.

7.35 C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_sdk_limits.h File Reference

Variables

- const uint32 [kMaxSubIFDs](#) = 10
The maximum number of SubIFDs that will be parsed.
- const uint32 [kMaxChainedIFDs](#) = 10
The maximum number of chained IFDs that will be parsed.

- const uint32 `kMaxSamplesPerPixel` = 4
The maximum number of samples per pixel.
- const uint32 `kMaxColorPlanes` = `kMaxSamplesPerPixel`
Maximum number of color planes.
- const uint32 `kMaxCFAPattern` = 8
The maximum size of a CFA repeating pattern.
- const uint32 `kMaxBlackPattern` = 8
The maximum size of a black level repeating pattern.
- const uint32 `kMaxMaskedAreas` = 4
The maximum number of masked area rectangles.
- const uint32 `kMaxImageSide` = 30000
The maximum image size supported (pixels per side).
- const uint32 `kMaxMPThreads` = 4
Maximum number of MP threads for `dng_area_task` operations.

7.35.1 Detailed Description

Collection of constants detailing maximum values used in processing in the DNG SDK.

8 DNG SDK Page Documentation

`dng_validate` Version 1.1 04-Apr-06

`§dng_validate` is a command-line tool that parses the tag structure of DNG (and other TIFF-EP based format) files, and reports any deviations from the DNG specification that it finds.

The usage syntax is:

```
dng_validate [-v] [-d <number>] [-f] [-b4] [-s < CFA index >] [-q <target-
binned-width>] [-cs1|-cs2|-cs3|-cs4|-cs5|-cs6] [-16] { [-1 <stage1-out-filename> ] [-
2 <stage2-out-filename> ] [-3 <stage3-out-filename> ] [-tif <TIFF-out-filename>]
[-dng <DNG-out-file>] {<list of files>}*
```

Any deviations from the DNG specification are written to the standard error stream.

The `$-v` option turns on `$verbose` mode, which writes the parsed tag structure to the standard output stream. Any tags that are not parsed by this tool are preceded by an asterisk.

The `$-d <number>` option both implies verbose mode, and also specifies the maximum number of lines of data displayed per tag.

The `$-f` option switches `dng_validate` to using floating-point math where possible, instead of the default 16-bit integer.

The `$-b4` option causes the demosaic algorithm to produce a four-channel output rather than a three-channel one. (The input DNG must be a three-channel Bayer pattern image.) This option is only useful when used with the `-3` switch. The extra channel is the result of doing two interpolations of the Bayer green channel such that the greens on the same row as the reds produces one channel and the greens on the same row as the blues produce another channel. The second green channel will be the highest numbered channel in the output. This option is used to gauge the difference between greens in each row to decide whether the DNG `BayerGreenSplit` tag should be used for a given source of image data (e.g. camera).

The `$-s <CFA index>` option chooses which set of color filter arrays to use when there are multiple ones for an input image. Each CFA array is a separate channel in the DNG input. This applies to the Fuji SR cameras for example, where the first channel is from the S-sensing elements and the second channel is from the R-sensing elements. The S elements are more sensitive and the R elements are less so with the goal of using both to increase the dynamic range the sensor can capture in a single image. By default `dng_validate` generates an image from only the S-sensors. By using `$-s 1` the R-sensing elements' data can be used to construct the output image. (This index is 0-based. The default is 0.)

The `$-q <target-binned-size>` option enables binning during the demosaic process. This is useful for creating previews or thumbnails. The binning factor is determined from the `target-binned-size`, which is the size in pixels of the larger dimension of the image that is desired. An integer binning factor will be computed to produce an image of that size or larger. For example, if the input image is 3008 x 2000 pixels and the `target-binned-width` is 700, factor of 4 binning will be used and the result output image (after demosaicing) will be 752 x 500 pixels.

The `$-cs1` option generates the output image in sRGB color space.

The `$-cs2` option generates the output image in AdobeRGB color space.

The `$-cs3` option generates the output image in ProPhotoRGB color space.

The `$-cs4` option generates the output image in ColorMatch color space.

The `$-cs5` option generates the output image in grayscale gamma 1.8 color space.

The `$-cs6` option generates the output image in a grayscale gamma 2.2 color space.

The `$-16` option causes `dng_validate` to output 16-bit-per-component images rather than the default 8-bit.

The `$-1` option causes the unprocessed raw image data to be written to the named output file. This applies only to the next input file after the switch.

The `$-2` option causes the image data after linearization and black/white level mapping to be written to the named output file. This applies only to the next input file after the switch.

The `$-3` option causes the image data after demosaic processing, but prior to color space conversion, noise reduction, sharpening, etc., to be written to the named output file. This applies only to the next input file after the switch.

The `$-tif` option causes the final rendered image to be written as TIFF to the named output file. This applies only to the next input file after the switch.

The `$-dng` option causes the parsed DNG data to be reserialized and written to the named output file. This mostly serves to provide an example code path for the process of writing a DNG file, as the output may not differ significantly from the input DNG. (Parameters, such as whether the data is compressed or not, may vary between the input and output DNG files.) This applies only to the next input file after the switch.

Index

- Allocate
 - dng_host, [59](#)
 - dng_memory_allocator, [79](#)
 - dng_memory_data, [85](#)
- ApplyOrientation
 - dng_negative, [103](#)
- AsMemoryBlock
 - dng_stream, [127](#)
- ASSERT
 - dng_assertions.h, [140](#)
- AutoPtr, [11](#)
 - AutoPtr, [12](#)
- AutoPtr
 - AutoPtr, [12](#)
- BestQualityFinalHeight
 - dng_negative, [103](#)
- BestQualityFinalWidth
 - dng_negative, [103](#)
- BigEndian
 - dng_stream, [127](#)
- BlackLevel
 - dng_linearization_info, [77](#)
- Buffer
 - dng_memory_block, [80](#)
 - dng_memory_data, [85](#)
- Buffer_char
 - dng_memory_block, [81](#)
 - dng_memory_data, [86](#)
- Buffer_int16
 - dng_memory_block, [81](#)
 - dng_memory_data, [86](#)
- Buffer_int32
 - dng_memory_block, [81](#)
 - dng_memory_data, [86](#)
- Buffer_real32
 - dng_memory_block, [82](#)
 - dng_memory_data, [87](#)
- Buffer_real64
 - dng_memory_block, [82](#)
 - dng_memory_data, [87](#)
- Buffer_uint16
 - dng_memory_block, [82](#)
 - dng_memory_data, [87](#)
- Buffer_uint32
 - dng_memory_block, [83](#)
 - dng_memory_data, [88](#)
- Buffer_uint8
 - dng_memory_block, [83](#)
 - dng_memory_data, [88](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_1d_function.h, [138](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_1d_table.h, [138](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_abort_sniffer.h, [139](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_area_task.h, [139](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_assertions.h, [140](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_auto_ptr.h, [140](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_bottlenecks.h, [141](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_camera_profile.h, [146](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_space.h, [146](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_color_spec.h, [147](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_date_time.h, [148](#)
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_

- errors.h, 149
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_exceptions.h, 150
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_exif.h, 152
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_fast_module.h, 152
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_file_stream.h, 152
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_filter_task.h, 152
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_fingerprint.h, 153
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_flags.h, 153
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_globals.h, 154
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_host.h, 154
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_ifd.h, 154
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_image.h, 155
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_image_writer.h, 155
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_info.h, 156
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_iptc.h, 157
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_linearization_info.h, 157
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_matrix.h, 157
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_memory_stream.h, 158
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_mosaic_info.h, 158
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_negative.h, 159
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_pixel_buffer.h, 159
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_read_image.h, 159
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_render.h, 160
- C:/Development/DigitalImaging/dng_sdk_1_1/dng_sdk/source/dng_sdk_limits.h, 160
- CalibrationIlluminant1
 - dng_camera_profile, 28
- CalibrationIlluminant2
 - dng_camera_profile, 28
- CameraToPCS
 - dng_color_spec, 33
- CameraWhite
 - dng_color_spec, 33
- ColumnBlack
 - dng_linearization_info, 77
- ConstPixel
 - dng_pixel_buffer, 106
- ConstPixel_int16
 - dng_pixel_buffer, 106
- ConstPixel_int32
 - dng_pixel_buffer, 106
- ConstPixel_int8
 - dng_pixel_buffer, 106
- ConstPixel_real32
 - dng_pixel_buffer, 107
- ConstPixel_uint16
 - dng_pixel_buffer, 107

- ConstPixel_uint32
 - dng_pixel_buffer, 107
- ConstPixel_uint8
 - dng_pixel_buffer, 108
- CopyArea
 - dng_pixel_buffer, 108
- CurrentDateTimeAndZone
 - dng_date_time.h, 149
- DefaultScale
 - dng_negative, 103
- DirtyPixel
 - dng_pixel_buffer, 108
- DirtyPixel_int16
 - dng_pixel_buffer, 109
- DirtyPixel_int32
 - dng_pixel_buffer, 109
- DirtyPixel_int8
 - dng_pixel_buffer, 109
- DirtyPixel_real32
 - dng_pixel_buffer, 110
- DirtyPixel_uint16
 - dng_pixel_buffer, 110
- DirtyPixel_uint32
 - dng_pixel_buffer, 110
- DirtyPixel_uint8
 - dng_pixel_buffer, 110
- dng_1d_concatenate, 13
 - dng_1d_concatenate, 14
 - Evaluate, 14
 - EvaluateInverse, 14
- dng_1d_function, 15
 - Evaluate, 16
 - EvaluateInverse, 16
- dng_1d_identity, 17
- dng_1d_table, 18
 - Initialize, 18
 - Interpolate, 19
- dng_abort_sniffer, 19
 - SniffForAbort, 20
 - StartTask, 20
 - UpdateProgress, 20
- dng_area_task, 21
 - FindTileSize, 22
 - Finish, 22
 - MaxThreads, 22
 - MaxTileSize, 22
 - MinTaskArea, 23
 - Perform, 23
 - Process, 23
 - ProcessOnThread, 24
 - RepeatingTile1, 24
 - RepeatingTile2, 24
 - RepeatingTile3, 25
 - Start, 25
 - UnitCell, 25
- dng_assertions.h
 - ASSERT, 140
- dng_camera_profile, 26
 - CalibrationIlluminant1, 28
 - CalibrationIlluminant2, 28
 - EqualData, 28
 - IsValid, 28
 - Name, 28
 - NameIsEmbedded, 28
 - SetCalibrationIlluminant1, 29
 - SetCalibrationIlluminant2, 29
 - SetCameraCalibration1, 29
 - SetCameraCalibration2, 29
 - SetColorMatrix1, 29
 - SetColorMatrix2, 30
 - SetName, 30
 - SetReductionMatrix1, 30
 - SetReductionMatrix2, 30
- dng_color_space, 30
 - ICCProfile, 32
- dng_color_spec, 32
 - CameraToPCS, 33
 - CameraWhite, 33
 - NeutralToXY, 33
 - SetWhiteXY, 33
 - WhiteXY, 33
- dng_color_spec.h
 - MapWhiteMatrix, 148
- dng_const_tile_buffer, 34
 - dng_const_tile_buffer, 34
- dng_date_time, 35
 - dng_date_time, 36
 - IsValid, 36
 - NotValid, 36
 - Parse, 36
- dng_date_time.h

- dng_date_time_format_exif, [149](#)
 - dng_date_time_format_unix_big_endian, [149](#)
 - dng_date_time_format_unix_little_endian, [149](#)
- dng_date_time.h
 - CurrentDateTimeAndZone, [149](#)
 - dng_date_time_format, [149](#)
 - LocalTimeZone, [149](#)
- dng_date_time_format
 - dng_date_time.h, [149](#)
- dng_date_time_format_exif
 - dng_date_time.h, [149](#)
- dng_date_time_format_unix_big_endian
 - dng_date_time.h, [149](#)
- dng_date_time_format_unix_little_endian
 - dng_date_time.h, [149](#)
- dng_date_time_storage_info, [37](#)
 - Format, [37](#)
 - IsValid, [37](#)
 - Offset, [38](#)
- dng_dirty_tile_buffer, [38](#)
 - dng_dirty_tile_buffer, [39](#)
- dng_exception, [39](#)
 - dng_exception, [39](#)
 - ErrorCode, [40](#)
- dng_exif, [40](#)
- dng_file_stream, [44](#)
 - dng_file_stream, [44](#)
- dng_filter_task, [45](#)
 - dng_filter_task, [46](#)
 - Process, [46](#)
 - ProcessArea, [46](#)
 - SrcArea, [47](#)
 - SrcTileSize, [47](#)
 - Start, [47](#)
- dng_fingerprint, [48](#)
- dng_function_exposure_ramp, [49](#)
 - Evaluate, [50](#)
- dng_function_exposure_tone, [50](#)
- dng_function_gamma_encode, [51](#)
 - Evaluate, [52](#)
- dng_function_GammaEncode_1_8, [52](#)
- dng_function_GammaEncode_1_8
 - Evaluate, [53](#)
 - EvaluateInverse, [53](#)
- dng_function_GammaEncode_2_2, [54](#)
 - dng_function_GammaEncode_2_2
 - Evaluate, [54](#)
 - EvaluateInverse, [55](#)
- dng_function_GammaEncode_sRGB, [55](#)
 - dng_function_GammaEncode_sRGB
 - Evaluate, [56](#)
 - EvaluateInverse, [56](#)
- dng_host, [57](#)
 - Allocate, [59](#)
 - dng_host, [59](#)
 - ForPreview, [59](#)
 - IsTransientError, [60](#)
 - PerformAreaTask, [60](#)
 - SetForPreview, [60](#)
 - SetKeepDNGPrivate, [60](#)
 - SetKeepOriginalFile, [60](#)
 - SetKeepStage1, [61](#)
 - SetKeepStage2, [61](#)
 - SetMinimumSize, [61](#)
 - SetNeedsImage, [61](#)
 - SetNeedsMeta, [61](#)
- dng_ifd, [62](#)
- dng_image
 - edge_none, [67](#)
 - edge_repeat, [67](#)
 - edge_repeat_zero_last, [67](#)
 - edge_zero, [67](#)
- dng_image, [65](#)
 - edge_option, [67](#)
 - Get, [67](#)
 - PixelRange, [67](#)
 - PixelSize, [67](#)
 - PixelType, [68](#)
 - Put, [68](#)
 - Rotate, [68](#)
 - Trim, [68](#)
- dng_image_writer, [68](#)
 - WriteDNG, [70](#)
 - WriteTIFF, [70](#)
- dng_info, [71](#)
 - IsValidDNG, [72](#)
 - Parse, [72](#)
- dng_iptc, [73](#)
 - IsEmpty, [74](#)

- NotEmpty, [74](#)
- Parse, [74](#)
- Spool, [75](#)
- dng_linearization_info, [75](#)
 - BlackLevel, [77](#)
 - ColumnBlack, [77](#)
 - fActiveArea, [78](#)
 - fLinearizationTable, [78](#)
 - fMaskedArea, [78](#)
 - Linearize, [77](#)
 - RowBlack, [77](#)
- dng_memory_allocator, [78](#)
 - Allocate, [79](#)
- dng_memory_block, [79](#)
 - Buffer, [80](#)
 - Buffer_char, [81](#)
 - Buffer_int16, [81](#)
 - Buffer_int32, [81](#)
 - Buffer_real32, [82](#)
 - Buffer_real64, [82](#)
 - Buffer_uint16, [82](#)
 - Buffer_uint32, [83](#)
 - Buffer_uint8, [83](#)
 - LogicalSize, [83](#)
- dng_memory_data, [84](#)
 - Allocate, [85](#)
 - Buffer, [85](#)
 - Buffer_char, [86](#)
 - Buffer_int16, [86](#)
 - Buffer_int32, [86](#)
 - Buffer_real32, [87](#)
 - Buffer_real64, [87](#)
 - Buffer_uint16, [87](#)
 - Buffer_uint32, [88](#)
 - Buffer_uint8, [88](#)
 - dng_memory_data, [85](#)
- dng_memory_stream, [89](#)
 - dng_memory_stream, [90](#)
- dng_mosaic_info, [90](#)
 - DownScale, [91](#)
 - DstSize, [92](#)
 - fBayerGreenSplit, [94](#)
 - fCFALayout, [94](#)
 - FullScale, [92](#)
 - Interpolate, [92](#)
 - InterpolateFast, [92](#)
 - InterpolateGeneric, [93](#)
 - IsColorFilterArray, [93](#)
 - SetFourColorBayer, [93](#)
- dng_negative, [94](#)
 - ApplyOrientation, [103](#)
 - BestQualityFinalHeight, [103](#)
 - BestQualityFinalWidth, [103](#)
 - DefaultScale, [103](#)
- dng_pixel_buffer, [104](#)
 - ConstPixel, [106](#)
 - ConstPixel_int16, [106](#)
 - ConstPixel_int32, [106](#)
 - ConstPixel_int8, [106](#)
 - ConstPixel_real32, [107](#)
 - ConstPixel_uint16, [107](#)
 - ConstPixel_uint32, [107](#)
 - ConstPixel_uint8, [108](#)
 - CopyArea, [108](#)
 - DirtyPixel, [108](#)
 - DirtyPixel_int16, [109](#)
 - DirtyPixel_int32, [109](#)
 - DirtyPixel_int8, [109](#)
 - DirtyPixel_real32, [110](#)
 - DirtyPixel_uint16, [110](#)
 - DirtyPixel_uint32, [110](#)
 - DirtyPixel_uint8, [110](#)
 - FlipH, [111](#)
 - FlipV, [111](#)
 - FlipZ, [111](#)
 - PixelRange, [111](#)
 - RepeatArea, [111](#)
 - RepeatPhase, [111](#)
 - SetZero, [112](#)
 - ShiftRight, [112](#)
- dng_render, [112](#)
 - dng_render, [113](#)
 - Exposure, [114](#)
 - FinalPixelType, [114](#)
 - FinalSpace, [114](#)
 - MaximumSize, [114](#)
 - Render, [114](#)
 - SetExposure, [114](#)
 - SetFinalPixelType, [115](#)
 - SetFinalSpace, [115](#)
 - SetMaximumSize, [115](#)
 - SetShadows, [115](#)

- SetToneCurve, 115
- SetWhiteXY, 115
- Shadows, 116
- ToneCurve, 116
- WhiteXY, 116
- dng_simple_image, 116
- dng_sniffer_task, 117
 - dng_sniffer_task, 118
 - UpdateProgress, 118
- dng_space_AdobeRGB, 119
- dng_space_ColorMatch, 120
- dng_space_GrayGamma18, 121
- dng_space_GrayGamma22, 122
- dng_space_ProPhoto, 123
- dng_space_sRGB, 123
- dng_stream, 124
 - AsMemoryBlock, 127
 - BigEndian, 127
 - dng_stream, 127
 - Get, 127
 - Get_CString, 127
 - Get_int16, 128
 - Get_int32, 128
 - Get_int8, 128
 - Get_real32, 129
 - Get_real64, 129
 - Get_uint16, 129
 - Get_uint32, 129
 - Get_uint8, 130
 - Get_UString, 130
 - Length, 130
 - LittleEndian, 130
 - OffsetInOriginalFile, 131
 - Position, 131
 - PositionInOriginalFile, 131
 - Put, 131
 - Put_int16, 131
 - Put_int32, 131
 - Put_int8, 132
 - Put_real32, 132
 - Put_real64, 132
 - Put_uint16, 132
 - Put_uint32, 132
 - Put_uint8, 133
 - SetBigEndian, 133
 - SetLength, 133
 - SetLittleEndian, 133
 - SetSwapBytes, 133
 - Skip, 133
 - Sniffer, 134
 - SwapBytes, 134
 - TagValue_int32, 134
 - TagValue_real64, 134
 - TagValue_srational, 135
 - TagValue_uint32, 135
 - TagValue_urational, 135
- dng_tile_buffer, 136
 - dng_tile_buffer, 137
- dng_tone_curve_acr3_default, 137
- DownScale
 - dng_mosaic_info, 91
- DstSize
 - dng_mosaic_info, 92
- edge_none
 - dng_image, 67
- edge_option
 - dng_image, 67
- edge_repeat
 - dng_image, 67
- edge_repeat_zero_last
 - dng_image, 67
- edge_zero
 - dng_image, 67
- EqualData
 - dng_camera_profile, 28
- ErrorCode
 - dng_exception, 40
- Evaluate
 - dng_1d_concatenate, 14
 - dng_1d_function, 16
 - dng_function_exposure_ramp, 50
 - dng_function_gamma_encode, 52
 - dng_function_GammaEncode_1_8, 53
 - dng_function_GammaEncode_2_2, 54
 - dng_function_GammaEncode_s-
RGB, 56
- EvaluateInverse
 - dng_1d_concatenate, 14
 - dng_1d_function, 16

- dng_function_GammaEncode_1_8,
53
- dng_function_GammaEncode_2_2,
55
- dng_function_GammaEncode_s-
RGB, 56
- Exposure
 - dng_render, 114
- fActiveArea
 - dng_linearization_info, 78
- fBayerGreenSplit
 - dng_mosaic_info, 94
- fCFALayout
 - dng_mosaic_info, 94
- FinalPixelType
 - dng_render, 114
- FinalSpace
 - dng_render, 114
- FindTileSize
 - dng_area_task, 22
- Finish
 - dng_area_task, 22
- fLinearizationTable
 - dng_linearization_info, 78
- FlipH
 - dng_pixel_buffer, 111
- FlipV
 - dng_pixel_buffer, 111
- FlipZ
 - dng_pixel_buffer, 111
- fMaskedArea
 - dng_linearization_info, 78
- Format
 - dng_date_time_storage_info, 37
- ForPreview
 - dng_host, 59
- FullScale
 - dng_mosaic_info, 92
- Get
 - dng_image, 67
 - dng_stream, 127
- Get_CString
 - dng_stream, 127
- Get_int16
 - dng_stream, 128
- Get_int32
 - dng_stream, 128
- Get_int8
 - dng_stream, 128
- Get_real32
 - dng_stream, 129
- Get_real64
 - dng_stream, 129
- Get_uint16
 - dng_stream, 129
- Get_uint32
 - dng_stream, 129
- Get_uint8
 - dng_stream, 130
- Get_UString
 - dng_stream, 130
- ICCProfile
 - dng_color_space, 32
- Initialize
 - dng_1d_table, 18
- Interpolate
 - dng_1d_table, 19
 - dng_mosaic_info, 92
- InterpolateFast
 - dng_mosaic_info, 92
- InterpolateGeneric
 - dng_mosaic_info, 93
- IsColorFilterArray
 - dng_mosaic_info, 93
- IsEmpty
 - dng_iptc, 74
- IsTransientError
 - dng_host, 60
- IsValid
 - dng_camera_profile, 28
 - dng_date_time, 36
 - dng_date_time_storage_info, 37
- IsValidDNG
 - dng_info, 72
- Length
 - dng_stream, 130
- Linearize
 - dng_linearization_info, 77

- LittleEndian
 - dng_stream, [130](#)
- LocalTimeZone
 - dng_date_time.h, [149](#)
- LogicalSize
 - dng_memory_block, [83](#)
- MapWhiteMatrix
 - dng_color_spec.h, [148](#)
- MaximumSize
 - dng_render, [114](#)
- MaxThreads
 - dng_area_task, [22](#)
- MaxTileSize
 - dng_area_task, [22](#)
- MinTaskArea
 - dng_area_task, [23](#)
- Name
 - dng_camera_profile, [28](#)
- NameIsEmbedded
 - dng_camera_profile, [28](#)
- NeutralToXY
 - dng_color_spec, [33](#)
- NotEmpty
 - dng_iptc, [74](#)
- NotValid
 - dng_date_time, [36](#)
- Offset
 - dng_date_time_storage_info, [38](#)
- OffsetInOriginalFile
 - dng_stream, [131](#)
- Parse
 - dng_date_time, [36](#)
 - dng_info, [72](#)
 - dng_iptc, [74](#)
- Perform
 - dng_area_task, [23](#)
- PerformAreaTask
 - dng_host, [60](#)
- PixelRange
 - dng_image, [67](#)
 - dng_pixel_buffer, [111](#)
- PixelSize
 - dng_image, [67](#)
- PixelType
 - dng_image, [68](#)
- Position
 - dng_stream, [131](#)
- PositionInOriginalFile
 - dng_stream, [131](#)
- Process
 - dng_area_task, [23](#)
 - dng_filter_task, [46](#)
- ProcessArea
 - dng_filter_task, [46](#)
- ProcessOnThread
 - dng_area_task, [24](#)
- Put
 - dng_image, [68](#)
 - dng_stream, [131](#)
- Put_int16
 - dng_stream, [131](#)
- Put_int32
 - dng_stream, [131](#)
- Put_int8
 - dng_stream, [132](#)
- Put_real32
 - dng_stream, [132](#)
- Put_real64
 - dng_stream, [132](#)
- Put_uint16
 - dng_stream, [132](#)
- Put_uint32
 - dng_stream, [132](#)
- Put_uint8
 - dng_stream, [133](#)
- Render
 - dng_render, [114](#)
- RepeatArea
 - dng_pixel_buffer, [111](#)
- RepeatingTile1
 - dng_area_task, [24](#)
- RepeatingTile2
 - dng_area_task, [24](#)
- RepeatingTile3
 - dng_area_task, [25](#)
- RepeatPhase
 - dng_pixel_buffer, [111](#)

- Rotate
 - dng_image, [68](#)
- RowBlack
 - dng_linearization_info, [77](#)
- SetBigEndian
 - dng_stream, [133](#)
- SetCalibrationIlluminant1
 - dng_camera_profile, [29](#)
- SetCalibrationIlluminant2
 - dng_camera_profile, [29](#)
- SetCameraCalibration1
 - dng_camera_profile, [29](#)
- SetCameraCalibration2
 - dng_camera_profile, [29](#)
- SetColorMatrix1
 - dng_camera_profile, [29](#)
- SetColorMatrix2
 - dng_camera_profile, [30](#)
- SetExposure
 - dng_render, [114](#)
- SetFinalPixelType
 - dng_render, [115](#)
- SetFinalSpace
 - dng_render, [115](#)
- SetForPreview
 - dng_host, [60](#)
- SetFourColorBayer
 - dng_mosaic_info, [93](#)
- SetKeepDNGPrivate
 - dng_host, [60](#)
- SetKeepOriginalFile
 - dng_host, [60](#)
- SetKeepStage1
 - dng_host, [61](#)
- SetKeepStage2
 - dng_host, [61](#)
- SetLength
 - dng_stream, [133](#)
- SetLittleEndian
 - dng_stream, [133](#)
- SetMaximumSize
 - dng_render, [115](#)
- SetMinimumSize
 - dng_host, [61](#)
- SetName
 - dng_camera_profile, [30](#)
- SetNeedsImage
 - dng_host, [61](#)
- SetNeedsMeta
 - dng_host, [61](#)
- SetReductionMatrix1
 - dng_camera_profile, [30](#)
- SetReductionMatrix2
 - dng_camera_profile, [30](#)
- SetShadows
 - dng_render, [115](#)
- SetSwapBytes
 - dng_stream, [133](#)
- SetToneCurve
 - dng_render, [115](#)
- SetWhiteXY
 - dng_color_spec, [33](#)
 - dng_render, [115](#)
- SetZero
 - dng_pixel_buffer, [112](#)
- Shadows
 - dng_render, [116](#)
- ShiftRight
 - dng_pixel_buffer, [112](#)
- Skip
 - dng_stream, [133](#)
- Sniffer
 - dng_stream, [134](#)
- SniffForAbort
 - dng_abort_sniffer, [20](#)
- Spool
 - dng_iptc, [75](#)
- SrcArea
 - dng_filter_task, [47](#)
- SrcTileSize
 - dng_filter_task, [47](#)
- Start
 - dng_area_task, [25](#)
 - dng_filter_task, [47](#)
- StartTask
 - dng_abort_sniffer, [20](#)
- SwapBytes
 - dng_stream, [134](#)
- TagValue_int32
 - dng_stream, [134](#)

- TagValue_real64
 - dng_stream, [134](#)
- TagValue_srational
 - dng_stream, [135](#)
- TagValue_uint32
 - dng_stream, [135](#)
- TagValue_urational
 - dng_stream, [135](#)
- ToneCurve
 - dng_render, [116](#)
- Trim
 - dng_image, [68](#)
- UnitCell
 - dng_area_task, [25](#)
- UpdateProgress
 - dng_abort_sniffer, [20](#)
 - dng_sniffer_task, [118](#)
- WhiteXY
 - dng_color_spec, [33](#)
 - dng_render, [116](#)
- WriteDNG
 - dng_image_writer, [70](#)
- WriteTIFF
 - dng_image_writer, [70](#)