Digital Negative (DNG) Specification

Version 1.7.1.0

September 2023

Adobe Inc.
345 Park Avenue
San Jose, CA 95110-2704
(408) 536-6000
http://www.adobe.com





Copyright © 2004 - 2023 Adobe Inc. All rights reserved.

NOTICE: All information contained herein is the property of Adobe Inc. No part of this publication (whether in hardcopy or electronic form) may be reproduced or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Adobe Inc.

Adobe, the Adobe logo, and Photoshop are either registered trademarks or trademarks of Adobe Inc. in the United States and/or other countries. All other trademarks are the property of their respective owners.

This publication and the information herein is furnished AS IS, is subject to change without notice, and should not be construed as a commitment by Adobe Inc. Adobe Inc. assumes no responsibility or liability for any errors or inaccuracies, makes no warranty of any kind (express, implied, or statutory) with respect to this publication, and expressly disclaims any and all warranties of merchantability, fitness for particular purposes, and noninfringement of third party rights.

Table of Contents

TABLE OF CONTENTS **3**

PREFACE **9**

About This Document 9
Audience 9
How This Document Is Organized 9
New Information for Version 1.7.0.0 10
New Information for Version 1.7.1.0 10
Where to Go for More Information 10

INTRODUCTION **11**

The Pros and Cons of Raw Data 11
A Standard Format 11
The Advantages of DNG 11

DNG FORMAT OVERVIEW **13**

File Extensions 13
SubIFD Trees 13
Byte Order 13
Masked Pixels 13
Defective Pixels 14
Metadata 14
Proprietary Data 14
Camera Profiles 14
Opcode Lists 15
Floating Point Image Data 16
Transparent Pixels 16
Proxy DNG Files 16
Depth Maps 16
Enhanced Image Data 17
Semantic Masks 17
64-bit Format 18

RESTRICTIONS AND EXTENSIONS TO EXISTING TIFF TAGS **20**

NewSubFileType 20
BitsPerSample 20
SampleFormat 20
Compression 20
Predictor 22
PhotometricInterpretation 22
Orientation 23

DNG TAGS **24**

DNGVersion 24
DNGBackwardVersion 24
UniqueCameraModel 25
LocalizedCameraModel 25
CFAPlaneColor 26
CFALayout 26
LinearizationTable 27
BlackLevelRepeatDim 27
BlackLevel 28
BlackLevelDeltaH 28
BlackLevelDeltaV 29
WhiteLevel 29
DefaultScale 30
BestQualityScale 30
DefaultCropOrigin 31
DefaultCropSize 31
CalibrationIlluminant1 31
CalibrationIlluminant2 32
ColorMatrix1 32
ColorMatrix2 33
CameraCalibration1 33
CameraCalibration2 34
ReductionMatrix1 34
ReductionMatrix2 35
AnalogBalance 35
AsShotNeutral 36
AsShotWhiteXY 36
BaselineExposure 36
BaselineNoise 37
BaselineSharpness 37
BayerGreenSplit 38
LinearResponseLimit 38
CameraSerialNumber 39
LensInfo 39
ChromaBlurRadius 39
AntiAliasStrength 40
ShadowScale 40
DNGPrivateData 41
MakerNoteSafety 41
RawDataUniqueID 42
OriginalRawFileName 42
OriginalRawFileData 42
ActiveArea 44
MaskedAreas 44

AsShotICCProfile 45
AsShotPreProfileMatrix 45
CurrentICCProfile 46
CurrentPreProfileMatrix 46
Additional Tags for Version 1.2.0.0 47
ColorimetricReference 47
CameraCalibrationSignature 47
ProfileCalibrationSignature 48
ExtraCameraProfiles 48
AsShotProfileName 48
NoiseReductionApplied 49
ProfileName 49
ProfileHueSatMapDims 49
ProfileHueSatMapData1 50
ProfileHueSatMapData2 50
ProfileToneCurve 51
ProfileEmbedPolicy 51
ProfileCopyright 52
ForwardMatrix1 52
ForwardMatrix2 53
PreviewApplicationName 53
PreviewApplicationVersion 53
PreviewSettingsName 54
PreviewSettingsDigest 54
PreviewColorSpace 54
PreviewDateTime 55
RawImageDigest 55
OriginalRawFileDigest 55
SubTileBlockSize 56
RowInterleaveFactor 56
ProfileLookTableDims 56
ProfileLookTableData 57
Additional Tags for Version 1.3.0.0 58
OpcodeList1 58
OpcodeList2 58
OpcodeList3 58
NoiseProfile 59
Additional Tags for Version 1.4.0.0 62
DefaultUserCrop 62
DefaultBlackRender 62
BaselineExposureOffset 63
ProfileLookTableEncoding 63
ProfileHueSatMapEncoding 64
OriginalDefaultFinalSize 65

OriginalBestQualityFinalSize 65
OriginalDefaultCropSize 66
NewRawImageDigest 66
RawToPreviewGain 66
Additional Tags for Version 1.5.0.0 68
DepthFormat 68
DepthNear 68
DepthFar 69
DepthUnits 69
DepthMeasureType 69
EnhanceParams 70
Additional Tags for Version 1.6.0.0 71
ProfileGainTableMap 71
SemanticName 73
SemanticInstanceID 73
MaskSubArea 74
RGBTables 75
CalibrationIlluminant3 81
ColorMatrix3 81
CameraCalibration3 82
ReductionMatrix3 82
ProfileHueSatMapData3 83
ForwardMatrix3 83
IlluminantData1 83
IlluminantData2 85
IlluminantData3 85
Additional Tags for Version 1.7.0.0 86
ProfileGainTableMap2 86
ImageSequenceInfo 88
ImageStats 90
ProfileDynamicRange 94
ProfileGroupName 96
Additional Tags for Version 1.7.1.0 97
ColumnInterleaveFactor 97
JXLDistance 97
JXLEffort 98
JXLDecodeSpeed 98

MAPPING RAW VALUES TO LINEAR REFERENCE VALUES 99

Linearization 99
Black Subtraction 99
Rescaling (Normalization) 99
Clipping 99

MAPPING CAMERA COLOR SPACE TO CIE XYZ SPACE 100

Camera Calibration Matrices 100
One, Two, or Three Color Calibrations 100
Definitions used in the following sections 101
Translating White Balance xy Coordinates to Camera Neutral Coordinates 101
Translating Camera Neutral Coordinates to White Balance xy Coordinates 102
Camera to XYZ (D50) Transform 102
If the ForwardMatrix tags are not included in the camera profile 102
If the ForwardMatrix tags are included in the camera profile 103
Applying the Hue/Saturation/Value Mapping Table 103
Special compatibility note with DNG 1.2 103

OPCODE LIST PROCESSING 105

WarpRectilinear 106
WarpFisheye 108
FixVignetteRadial 110
FixBadPixelsConstant 111
FixBadPixelsList 112
TrimBounds 113
MapTable 113
MapPolynomial 114
GainMap 115
DeltaPerRow 116
DeltaPerColumn 117
ScalePerRow 118
ScalePerColumn 119
WarpRectilinear2 120

APPENDIX A: COMPATIBILITY WITH PREVIOUS VERSIONS 123

Compatibility Issue 1: ActiveArea Tag 123
Compatibility Issue 2: 16-bit Lossless JPEG Encoding 123
Compatibility Issue 3: SubTileBlockSize 123
Compatibility Issue 4: RowInterleaveFactor 124
Compatibility Issue 5: PreviewColorSpace 124
Compatibility Issue 6: CFALayout 124
Compatibility Issue 7: Opcode Lists 124
Compatibility Issue 8: Floating Point Image Data 124
Compatibility Issue 9: Transparent Pixels 124
Compatibility Issue 10: Deflate Compression 124
Compatibility Issue 11: Lossy JPEG Compression 125
Compatibility Issue 12: Depth Maps 125
Compatibility Issue 13: Enhanced Image Data 125
Compatibility Issue 14: Semantic Mask 125
Compatibility Issue 15: ProfileGainTableMap 125
Compatibility Issue 16: WarpRectilinear2 Opcode 125

Compatibility Issue 17: Third Color Calibration and Custom Illuminant Data 125
Compatibility Issue 18: JPEG XL Compression 125
Compatibility Issue 19: ColumnInterleaveFactor 125
Compatibility Issue 20: ProfileGainTableMap and Camera Profile IFD 126

Preface

About This Document

The Digital Negative (DNG) Specification describes a non-proprietary file format for storing camera raw files that can be used by a wide range of hardware and software vendors.

This section contains information about this document, including how it is organized and where to go for additional information.

Audience

This document is intended for developers of hardware and software applications that will generate, process, manage, or archive camera raw files.

How This Document Is Organized

This document has the following sections:

Chapter 1, "Introduction" explains what digital negatives are, gives an overview of the DNG file format, and discusses the advantages of DNG.

Chapter 2, "DNG Format Overview" provides an overview of the DNG format, including information on file extensions, SubIFD trees, byte order, masked pixels, defective pixels, metadata, and proprietary data.

Chapter 3, "Restrictions and Extensions to Existing TIFF Tags" describes tag differences between DNG and the TIFF 6.0 format on which DNG is based.

Chapter 4, "DNG Tags" lists all DNG-specific tags and describes how they are used.

Chapter 5, "Mapping Raw Values to Linear Reference Values" specifies DNG's processing model for mapping stored raw sensor values into linear reference values.

Chapter 6, "Mapping Camera Color Space to CIE XYZ Space" describes DNG's processing model for mapping between the camera color space coordinates (linear reference values) and CIE XYZ coordinates.

Chapter 7, "Opcode List Processing" describes the concept of "Opcode Lists", which allow additional processing steps to be specified in an extensible manner.

"Appendix A: Compatibility with Previous Versions" documents compatibility between the current and previous DNG versions.

New Information for Version 1.7.0.0

The following changes have been made for the 1.7.0.0 version of this specification:

- A section on JPEG XL compression has been added to Chapter 2, "DNG Format Overview."
- The section on Compression has been updated in Chapter 3, "Restrictions and Extensions to Existing TIFF Tags."
- Additional tags were added to Chapter 4, "DNG Tags" for Version 1.7.0.0.
- The ColorimetricReference tag was updated with a new value for HDR output-referred images.
- "Appendix A: Compatibility with Previous Versions" was updated.

New Information for Version 1.7.1.0

The following changes have been made for the 1.7.1.0 version of this specification:

- The ColumnInterleaveFactor tag was moved from version 1.7.0.0 to version 1.7.1.0.
- Additional tags were added to Chapter 4, "DNG Tags" for Version 1.7.1.0.
- "Appendix A: Compatibility with Previous Versions" was updated.

Where to Go for More Information

DNG is an extension of TIFF 6.0 and is compatible with the TIFF-EP standard. See these specifications for more information on TIFF and TIFF-EP:

TIFF 6.0 Specification, Adobe Systems, Inc., 1992-06-03.	https://www.adobe.io/content/dam/udp/en/open/standards/tiff/TIFF6.pdf
TIFF/EP Specification, ISO/DIS 12234-2, 2001-10-15.	http://www.iso.org
Adobe Photoshop® TIFF Technical Notes	https://www.adobe.io/content/dam/udp/en/open/standards/tiff/TIFFphotoshop.pdf
Adobe Photoshop® TIFF Technical Note 3	http://chriscox.org/TIFFTN3d1.pdf

Introduction

The Pros and Cons of Raw Data

Seeking a greater degree of flexibility and artistic control, professional photographers increasingly opt to manipulate raw data from their digital cameras. Unlike JPEG and TIFF formats which store images that have been processed by the camera, camera raw files capture unprocessed or minimally processed data directly from the camera sensor. Because they are analogous to film negatives in a photographer's workflow, camera raw formats are often referred to as "digital negatives."

Camera raw formats offer both advantages and disadvantages. One advantage is increased artistic control for the end user. The user can precisely adjust a range of parameters, including white balance, tone mapping, noise reduction, sharpening and others, to achieve a desired look.

One disadvantage is that unlike JPEG and TIFF files which are ready for immediate use, camera raw files must be processed before they can be used, typically through software provided by the camera manufacturer or through a converter like the Camera Raw plug-in for Adobe Photoshop® software.

The challenge for end users and camera vendors alike is that there is no publicly-documented and supported format for storing raw camera data. Every camera manufacturer that supports raw data must create their own proprietary format, along with software for converting the proprietary format into the standard JPEG and/or TIFF formats.

A Standard Format

The lack of a standard format for camera raw files creates additional work for camera manufacturers because they need to develop proprietary formats along with the software to process them. It also poses risks for end users. Camera raw formats vary from camera to camera, even those produced by the same manufacturer. It is not uncommon for a camera manufacturer to terminate support for a discontinued camera's raw format. This means users have no guarantee they will be able to open archived camera raw files in the future.

To address these problems, Adobe has defined a new non-proprietary format for camera raw files. The format, called Digital Negative or DNG, can be used by a wide range of hardware and software developers to provide a more flexible raw processing and archiving workflow. End users may also use DNG as an intermediate format for storing images that were originally captured using a proprietary camera raw format.

The Advantages of DNG

DNG has all the benefits of current camera raw formats; namely, increased flexibility and artistic control. In addition, DNG offers several new advantages over proprietary camera raw formats.

Self-Contained

With the current proprietary camera raw formats, software programs wishing to process camera raw files must have specific information about the camera that created the file. As new camera models are released, software manufacturers (and by extension users) must update their software to accommodate the new camera raw formats.

Because DNG metadata is publicly documented, software readers such as the Adobe Photoshop Camera Raw plug-in do not need camera-specific knowledge to decode and process files created by a camera that supports DNG. That means reduced software maintenance and a more self-contained solution for end users.

Archival

Camera manufacturers sometimes drop support for a propriety raw format a few years after a camera is discontinued. Without continued software support, users may not be able to access images stored in proprietary raw formats and the images may be lost forever. Since DNG is publicly documented, it is far more likely that raw images stored as DNG files will be readable by software in the distant future, making DNG a safer choice for archival.

TIFF Compatible

DNG is an extension of the TIFF 6.0 format, and is compatible with the TIFF-EP standard. It is possible (but not required) for a DNG file to simultaneously comply with both the Digital Negative specification and the TIFF-EP standard.

DNG Format Overview

This section describes the DNG format. As an extension of the TIFF 6.0 format, DNG should follow all the formatting rules for TIFF 6.0. For more information, refer to the TIFF 6.0 specification.

The following topics are discussed in this section:

- File Extensions
- SubIFD Trees
- Byte Order
- Masked Pixels
- Defective Pixels
- Metadata
- Proprietary Data
- Camera Profiles
- Opcode Lists
- Floating Point Image Data
- Transparent Pixels
- Proxy DNG Files
- Depth Maps
- Enhanced Image Data

File Extensions

The recommended file extension for Digital Negative is ".DNG". Readers should accept either the ".DNG" or ".TIF" extensions for compatibility with TIFF-EP.

SubIFD Trees

DNG recommends the use of SubIFD trees, as described in the TIFF-EP specification. SubIFD chains are not supported.

The highest-resolution and quality IFD should use NewSubFileType equal to 0. Reduced resolution (or quality) thumbnails or previews, if any, should use NewSubFileType equal to 1 (for a primary preview) or 10001.H (for an alternate preview).

DNG recommends, but does not require, that the first IFD contain a low-resolution thumbnail, as described in the TIFF-EP specification.

Byte Order

DNG readers are required to support either byte order, even for files from a particular camera model. Writers can write either byte order, whichever is easier and/or faster for the writer.

Masked Pixels

Most camera sensors measure the black encoding level using fully-masked pixels at the edges of the sensor. These pixels can either be trimmed before storing the image in DNG, or they can be included in the stored image. If the masked pixels are not trimmed, the area of the non- masked pixels must be specified using the ActiveArea tag.

The black encoding level information extracted from these masked pixels should be used to either pre-compensate the raw data stored in the file or they should be included in the file using the DNG tags for specifying the black level.

This black encoding level information is required even if the masked pixels are not trimmed, to allow DNG readers to process the image without requiring knowledge of the best way to compute the black levels for any given camera model.

Defective Pixels

There are two ways to deal with defective pixels in DNG. The first is to map out (interpolate over) the defective pixels before storing the raw data in DNG. The second is to include a bad pixel fixing opcode in the OpcodeList1 tag.

Metadata

Additional metadata may be embedded in DNG in the following ways:

- Using TIFF-EP or EXIF metadata tags
- Using the IPTC metadata tag (33723)
- Using the XMP metadata tag (700)

Note that TIFF-EP and EXIF use nearly the same metadata tag set, but TIFF-EP stores the tags in IFD 0, while EXIF store the tags in a separate IFD. Either location is allowed by DNG, but the EXIF location is preferred.

Proprietary Data

Camera manufacturers may want to include proprietary data in a raw file for use by their own raw converter. DNG allows proprietary data to be stored using private tags, private IFDs, and/or a private MakerNote.

It is recommended that manufacturers use the DNGPrivateData and MakerNoteSafety tags to ensure that programs that edit DNG files preserve this proprietary data. See Chapter 4, “DNG Tags” on page 24 for more information on the DNGPrivateData and MakerNoteSafety tags.

Camera Profiles

DNG 1.2.0.0 and later formalizes the concept of a “camera profile” and allows multiple camera profiles to be embedded in a single DNG file. A camera profile consists of a set of tags (both existing in DNG versions earlier than 1.2.0.0 and newly defined in DNG version 1.2.0.0), some of which are optional.

The set of tags belonging to a camera profile includes the following:

- BaselineExposureOffset
- CalibrationIlluminant1
- CalibrationIlluminant2
- CalibrationIlluminant3†
- ColorMatrix1
- ColorMatrix2
- ColorMatrix3†
- DefaultBlackRender
- ForwardMatrix1
- ForwardMatrix2

- ForwardMatrix3†
- IlluminantData1†
- IlluminantData2†
- IlluminantData3†
- ProfileCalibrationSignature
- ProfileCopyright
- ProfileDynamicRange††
- ProfileEmbedPolicy
- ProfileGroupName††
- ProfileHueSatMapData1
- ProfileHueSatMapData2
- ProfileHueSatMapData3†
- ProfileHueSatMapDims
- ProfileHueSatMapEncoding
- ProfileLookTableData
- ProfileLookTableDims
- ProfileLookTableEncoding
- ProfileName
- ProfileToneCurve
- ReductionMatrix1
- ReductionMatrix2
- ReductionMatrix3†

The primary camera profile is stored in IFD 0, as it was for DNG versions earlier than 1.2.0.0. This allows backward compatibility with older DNG readers. DNG allows additional camera profiles to be embedded using the ExtraCameraProfiles tag, which points to a list of Camera Profile IFDs.

† Note that the profile tags describing a third illuminant (CalibrationIlluminant3, ColorMatrix3, ForwardMatrix3, ProfileHueSatMapData3, and ReductionMatrix3) and for specifying custom illuminant data (IlluminantData1, IlluminantData2, and IlluminantData3) are new to DNG 1.6.0.0.

†† Note that the profile tags ProfileDynamicRange and ProfileGroupName are new to DNG 1.7.0.0. ProfileDynamicRange introduces the concept of High Dynamic Range camera profiles, where an encoding and decoding function need to be used to apply tone curves and lookup tables (which are usually defined on a normalized range [0, 1]) to overrange pixel values.

Opcode Lists

DNG 1.3.0.0 and later incorporates the concept of “Opcode Lists”, which allow additional processing steps to be specified in an extensible manner.

This allows complex processing to be moved off the camera hardware, which often has limited processing power, and into the DNG reader, which is often running on more powerful hardware.

This also allows processing steps to be specified, such as lens corrections, which ideally should be performed on the image data after it has been demosaiced, while still retaining the advantages of a raw mosaic data format.

The set of tags controlling this feature are:

- OpcodeList1
- OpcodeList2
- OpcodeList3

Floating Point Image Data

DNG 1.4.0.0 and later support floating point image data, in addition to the previous support for unsigned integer image data.

Floating point bit depths of 16, 24 and 32 bits per sample are supported.

Transparent Pixels

DNG 1.4.0.0 and later incorporates the concept of "Transparent Pixels", which allow the level of transparency of pixels to be specified for the image data. This is useful when not all of the pixels in the image bounding rectangle are defined. For example, after combining several source images into a panorama, the edge of the resulting combined image is often not perfectly rectangular.

Transparency is supported by allowing masks to be stored in IFDs with the same width and length as the image IFDs, but with a NewSubFileType of 4 or 5 (see the section on NewSubFileType for details).

Fully transparent (undefined) pixels are stored with a mask value of zero. Fully opaque (defined) pixels are stored with a mask value of 1.0 for floating point masks, and the maximum possible unsigned integer for integer masks. Intermediate values are allowed for semi-transparent pixels.

For semi-transparent pixels, the image values should not be matted to either white or black, but instead stored with the same value as if the pixel was fully opaque.

Proxy DNG Files

DNG 1.4.0.0 adds the concept of proxy DNG files, which are lower resolution (and/or quality) versions of original DNG files. These proxy DNGs are useful files as placeholders for original DNGs in applications that have limited bandwidth or available storage space.

Proxy DNGs have information on the pixel size of the original DNG file to allow rendering parameters to be transferred back to the original DNGs with similar visual results.

The size of the corresponding original DNG file is specified using the OriginalDefaultFinalSize, OriginalBestQualitySize and OriginalCropSize tags.

Also added in DNG 1.4.0.0 and DNG 1.7.0.0 are new lossy compression options (JPEG and JPEG XL, respectively) that are particularity useful for proxy DNGs, greatly reducing their file size.

Depth Maps

DNG 1.5.0.0 and later incorporates the concept of "Depth Maps", which allow estimated or measured depth information to be specified for the image data.

Depth information is supported by allowing depth maps to be stored in IFDs with a NewSubFileType of 8 or 9 (see section on NewSubFileType for details).

The depth information IFD should use a PhotometricInterpretation of 51177 (see section on PhotometricInterpretation for details).

Depth information is stored as an 8 or 16-bit single channel unsigned integer image. Zero encodes the “nearest distance” (closest to the camera), and the maximum possible unsigned integer encodes the “farthest distance”. The mapping of the depth map integer values to distance values is specified by the DepthFormat, DepthNear, DepthFar, DepthUnits, and DepthMeasureType tags.

The depth map should match the field of view of the main image. It may have a different pixel resolution than the main image, but if the depth map is resized to match the size of the main image, the pixels should correspond. The depth values should be filled in by an estimation algorithm for any areas of the depth map that cannot be measured directly (i.e. no holes are allowed in depth maps).

Enhanced Image Data

DNG 1.5.0.0 adds the concept of enhanced image data, which is a more processed (and always at least demosaiced) version of the raw image data.

Enhanced image data is stored in an IFD with a NewSubFileType of 16 (see section on NewSubFileType for details).

The enhanced image data should always use a PhotometricInterpretation of 34892 (LinearRaw). It should use the same color space as the raw image data.

The enhanced image data should use either 16-bit unsigned integer encoding, or 16 or 24 or 32-bit floating point encoding.

The enhanced image data is assumed to have already been mapped to linear reference values, with the exception that it may have a non-zero BlackLevel tag if unsigned integer encoding is used. This black level is required to be the same for all color planes of the enhanced image data.

The enhanced IFD may have its own copies of the BaselineSharpness, NoiseProfile, and NoiseReductionApplied tags, which may differ from the raw IFD versions of these tags due to sharpening or noise reduction having been applied to the enhanced image data.

The enhanced IFD should have a non-null EnhanceParams tag to document the operations used to enhance the image data.

Semantic Masks

DNG 1.6.0.0 and later supports the concept of "Semantic Masks" which may be used for purposes such as image segmentation.

Semantic information is supported by allowing semantic masks (images) to be stored in IFDs with a NewSubFileType of 65540 (10004.H). See section on NewSubFileType for details.

The semantic information IFD should use a PhotometricInterpretation of 52527 (CD2F.H). See section on PhotometricInterpretation for details.

Semantic information is stored as a single channel image with one of the following pixel data types, with valid encoding range indicated in parentheses:

- 8-bit unsigned integer (0 to 255)
- 16-bit unsigned integer (0 to 65535)

- 32-bit floating point (0.0 to 1.0)

The minimum encoding value (zero) represents "not a part of the semantic attribute."

The maximum encoding value represents "is a part of the semantic attribute."

Encoding values in between the minimum and maximum values are allowed; they represent a partial semantic attribute. These fractional values may be used for matting or blending purposes.

A semantic mask should match the field of view of the main image, as specified by the ActiveArea tag. It may have a different (usually lower) pixel resolution than the main image.

Semantic masks may be stored uncompressed, or compressed with one of the following methods:

Compression Method	Required Value for Compression Tag	Data Types Supported
Lossless JPEG	7 decimal (7.H)	8-bit and 16-bit unsigned integer
Baseline DCT (lossy) JPEG	34892 decimal (884C.H)	8-bit unsigned integer
Deflate / Zip	8 (8.H)	8-bit and 16-bit unsigned int 32-bit float
JPEG XL	52546 decimal (CD42.H)	8-bit and 16-bit unsigned int 16-bit float

Note that JPEG XL requires DNG 1.7.0.0 or later.

Multiple semantic masks may be stored in a DNG file. Each semantic mask IFD should include a SemanticName tag with a unique string that identifies the purpose of the mask. See tag SemanticName for details.

A semantic mask should have the same orientation as the main image.

64-bit Format

TIFF format was designed using 32-bit values to store file offsets, which limits the maximum size of a TIFF file to 4 gigabytes. There is a 64-bit extension of TIFF (known as BigTIFF) which does not have this file size limitation. The DNG format allows this same 64-bit extension. Note that 64-bit format support is independent of the DNG version.

The differences from 32-bit DNG to 64-bit DNG are:

1. The 16-bit value at file offset 2 is the constant 43 (instead of 42).
2. The 16-bit value at file offset 4 is the constant 8.
3. The 16-bit value at file offset 6 is the constant 0.

4. The 64-bit value at file offset 8 is the offset to the first image file directory.
5. The entry count for each image file directory is a 64-bit value (instead of 16).
6. The element count for an image file directory entry is a 64-bit value (instead of 32).
7. The data entry for each image file directory entry is a 64-bit value (instead of 32).
8. The offset to the next image file directory is a 64-bit value (instead of 32).

Also, three new data types have been defined for image file directory entries:

- 16, which is an unsigned 64-bit integer.
- 17, which is a signed 64-bit integer.
- 18, which is a 64-bit file offset for an IFD.

It is recommended that DNG writers only use the 64-bit extension of DNG when the resulting file would be larger than 4 gigabytes to maximize compatibility with existing DNG readers.

Restrictions and Extensions to Existing TIFF Tags

This section describes the restrictions and extension to the following TIFF tags:

- NewSubFileType
- BitsPerSample
- SampleFormat
- Compression
- Predictor
- PhotometricInterpretation
- Orientation

NewSubFileType

In DNG versions earlier than 1.2.0.0, full resolution raw images should use NewSubFileType equal to 0. Rendered previews or reduced resolution versions of raw images should use NewSubFileType equal to 1. DNG 1.2.0.0 allows a new value for NewSubFileType equal to 10001.H.

This value, used for alternative or non-primary rendered previews, allows for multiple renderings (not just multiple sizes of a single rendering) to be stored in a DNG file. DNG reading software that displays a preview for a DNG file should, by default, display a preview from an IFD with NewSubFileType equal to 1. Alternative renderings should only be displayed if requested by the user.

DNG 1.4.0.0 allows two new values for NewSubFileType. Transparency information for the full resolution raw image should use NewSubFileType equal to 4. Transparency information for reduced resolution versions of raw images should use NewSubFileType equal to 5.

DNG 1.5.0.0 allows three new values for NewSubFileType. Depth map information of the highest resolution should use NewSubFileType equal to 8. Depth map information of lower resolutions should use a NewSubFileType of 9. Enhanced image data should use a NewSubFileType equal to 16.

DNG 1.6.0.0 allows a new value of NewSubFileType. Semantic mask information should use NewSubFileType equal to 65540 (10004.H). Multiple semantic masks can be stored in a DNG file.

BitsPerSample

Supported values are from 8 to 32 bits/sample. The depth must be the same for each sample if SamplesPerPixel is not equal to 1. If BitsPerSample is not equal to 8 or 16 or 32, then the bits must be packed into bytes using the TIFF default FillOrder of 1 (big-endian), even if the TIFF file itself uses little-endian byte order.

SampleFormat

DNG versions earlier than 1.4.0.0 always use the default SampleFormat value of 1 (unsigned integer). DNG version 1.4.0.0 allows an additional SampleFormat value of 3 (IEEE floating point). The allowed BitsPerSample values for floating point images are 16, 24, and 32.

Compression

Two Compression tag values are supported in DNG versions before 1.4.0.0:

- Value = 1: Uncompressed data.
- Value = 7: JPEG compressed data, either baseline DCT JPEG, or lossless JPEG compression.

If PhotometricInterpretation = 6 (YCbCr) and BitsPerSample = 8/8/8, or if PhotometricInterpretation = 1 (BlackIsZero) and BitsPerSample = 8, then the JPEG variant must be baseline DCT JPEG.

Otherwise, the JPEG variant must be lossless Huffman JPEG. For lossless JPEG, the internal width/length/components in the JPEG stream are not required to match the strip or tile's width/length/components. Only the total sample counts need to match. It is common for CFA images to be encoded with a different width, length or component count to allow the JPEG compression predictors to work across like colors.

DNG Version 1.4.0.0 adds support for the following compression codes:

- Value = 8: Deflate (ZIP)
- Value = 34892: Lossy JPEG

Deflate (8) compression is allowed for floating point image data, 32-bit integer image data, transparency mask data, and depth map data.

Lossy JPEG (34892) is allowed for IFDs that use 8-bit integer data and one of the following PhotometricInterpretation values:

- 34892 (LinearRaw)
- 52527 (PhotometricMask)

This compression code is required to let the DNG reader know to use a lossy JPEG decoder rather than a lossless JPEG decoder for this combination of PhotometricInterpretation and BitsPerSample.

DNG Version 1.7.0.0 adds support for JPEG XL compression (see ISO/IEC 18181-1:2022):

- Value = 52546: JPEG XL

JPEG XL compression is allowed for N-bit integer image data, where $8 \leq N \leq 16$, and 16-bit floating point image data. The number of images planes must be either 1 or 3. The PhotometricInterpretation must be one of the following values:

- 0 (WhiteIsZero)
- 1 (BlackIsZero)
- 2 (RGB)
- 4 (TransparencyMask)
- 32803 (ColorFilterArray)
- 34892 (LinearRaw)
- 51177 (Depth)
- 52527 (PhotometricMask)

JPEG XL supports two types of bitstreams:

- A “bare” format with no metadata
- A “container” format that uses the ISO Base Media File Format box structure with optional metadata

Both types of JPEG XL bitstreams are supported in DNG.

When using JPEG XL with multiple tiles, it is recommended to use the bare format, because each compressed tile will be slightly smaller, and per-tile metadata is not required.

When using JPEG XL to represent a preview IFD, such as a thumbnail or full-resolution rendered preview, it is recommended to use a single tile with the container format, so that utilities can extract a readable, standalone JPEG XL file by simply copying the bytes from the IFD.

Predictor

The following predictor codes are allowed for IFDs using compression code 8 (deflate):

- 1 = Null Predictor. No predictor is used as part of the compression.
- 2 = Horizontal Difference. Documented in the TIFF 6.0 specification.
- 3 = Floating Point. Documented in “Adobe Photoshop TIFF Technical Note 3”.
- 34892 = Horizontal Difference X2. Same as Horizontal Difference except the pixel two to the left is used rather than the pixel one to the left.
- 34893 = Horizontal Difference X4. Same as Horizontal Difference except the pixel four to the left is used rather than the pixel one to the left.
- 34894 = Floating Point X2. Same as Floating Point except the pixel two to the left is used rather than the pixel one to the left.
- 34895 = Floating Point X4. Same as Floating Point except the pixel four to the left is used rather than the pixel one to the left.

PhotometricInterpretation

The following values are supported for thumbnail and preview IFDs only:

- 1 = BlackIsZero. Assumed to be in a gamma 2.2 color space, unless otherwise specified using PreviewColorSpace tag.
- 2 = RGB. Assumed to be in the sRGB color space, unless otherwise specified using the PreviewColorSpace tag.
- 6 = YCbCr. Used for JPEG encoded preview images.

The following values are supported for the raw IFD, and are assumed to be the camera's native color space:

- 32803 = CFA (Color Filter Array).
- 34892 = LinearRaw.

The CFA PhotometricInterpretation value is documented in the TIFF-EP specification. Its use requires the use of the CFARepeatPatternDim and CFAPattern tags in the same IFD. The origin of the repeating CFA pattern is the top-left corner of the ActiveArea rectangle.

The LinearRaw PhotometricInterpretation value is intended for use by cameras that do not use color filter arrays, but instead capture all color components at each pixel. It can also be used for CFA data that has already been de-mosaiced.

The LinearRaw value can be used in reduced resolution IFDs and enhanced IFDs, even if the raw IFD uses the CFA PhotometricInterpretation value.

The following value is required for depth map IFDs:

- 51177 = Depth

The following value is required for semantic mask IFDs:

- 52527 = PhotometricMask

Orientation

Orientation is a required tag for DNG. With the Orientation tag present, file browsers can perform lossless rotation of DNG files by modifying a single byte of the file. DNG readers should support all possible orientations, including mirrored orientations. Note that the mirrored orientations are not allowed by the TIFF-EP specification, so writers should not use them if they want their files be compatible with both specifications.

DNG Tags

This section describes DNG-specific tags. Note that the tags listed here are not part of the TIFF-EP specification.

DNGVersion

Tag 50706 (C612.H)

Type BYTE

Count 4

Value See below

Default Required tag

Usage IFD 0

Description

This tag encodes the DNG four-tier version number. For files compliant with this version of the DNG specification (1.7.0.0), this tag should contain the bytes: 1, 7, 0, 0.

DNGBackwardVersion

Tag 50707 (C613.H)

Type BYTE

Count 4

Value See below

Default DNGVersion with the last two bytes set to zero

Usage IFD 0

Description

This tag specifies the oldest version of the Digital Negative specification for which a file is compatible. Readers should not attempt to read a file if this tag specifies a version number that is higher than the version number of the specification the reader was based on.

In addition to checking the version tags, readers should, for all tags, check the types, counts, and values, to verify it is able to correctly read the file.

For more information on compatibility with previous DNG versions, see Appendix A: Compatibility with Previous Versions.

UniqueCameraModel

Tag 50708 (C614.H)

Type ASCII

Count String length including null

Value Null-terminated string

Default Required tag

Usage IFD 0

Description

UniqueCameraModel defines a unique, non-localized name for the camera model that created the image in the raw file. This name should include the manufacturer's name to avoid conflicts, and should not be localized, even if the camera name itself is localized for different markets (see LocalizedCameraModel).

This string may be used by reader software to index into per-model preferences and replacement profiles.

Examples of unique model names are:

- "Canon EOS 300D"
- "Fujifilm FinePix S2Pro"
- "Kodak ProBack645"
- "Minolta DiMAGE A1"
- "Nikon D1X"
- "Olympus C-5050Z"
- "Pentax *istD"
- "Sony F828"

LocalizedCameraModel

Tag 50709 (C615.H)

Type ASCII or BYTE

Count Byte count including null

Value Null-terminated UTF-8 encoded Unicode string

Default Same as UniqueCameraModel

Usage IFD 0

Description

Similar to the UniqueCameraModel field, except the name can be localized for different markets to match the localization of the camera name.

CFAPlaneColor

Tag 50710 (C616.H)

Type BYTE

Count ColorPlanes

Value See below

Default 0, 1, 2 (red, green blue)

Usage Raw IFD

Description

CFAPlaneColor provides a mapping between the values in the CFAPattern tag and the plane numbers in LinearRaw space. This is a required tag for non-RGB CFA images.

CFALayout

Tag 50711 (C617.H)

Type SHORT

Count 1

Value See below

Default 1

Usage Raw IFD

Description

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
- 6 = Staggered layout E: even rows are offset up by 1/2 row, even columns are offset left by 1/2 column
- 7 = Staggered layout F: even rows are offset up by 1/2 row, even columns are offset right by 1/2 column
- 8 = Staggered layout G: even rows are offset down by 1/2 row, even columns are offset left by 1/2 column
- 9 = Staggered layout H: even rows are offset down by 1/2 row, even columns are offset right by 1/2 column

Note that for the purposes of this tag, rows and columns are numbered starting with one.

Layout values 6 through 9 were added with DNG version 1.3.0.0.

LinearizationTable

Tag 50712 (C618.H)

Type SHORT

Count N

Value See below

Default Identity table (0, 1, 2, 3, etc.)

Usage Raw IFD

Description

LinearizationTable describes 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, this single table applies to all the samples for each pixel.

See Chapter 5, "Mapping Raw Values to Linear Reference Values" on page 99 for details of the processing model.

BlackLevelRepeatDim

Tag 50713 (C619.H)

Type SHORT

Count 2

Value Value 0: BlackLevelRepeatRows

Value 1: BlackLevelRepeatCols

Default 1, 1

Usage Raw IFD

Description

This tag specifies repeat pattern size for the BlackLevel tag.

BlackLevel

Tag 50714 (C61A.H)

Type SHORT or LONG or RATIONAL

Count BlackLevelRepeatRows * BlackLevelRepeatCols * SamplesPerPixel

Value See below

Default 0

Usage Raw IFD

Description

This tag specifies the zero light (a.k.a. thermal black or black current) encoding level, as a repeating pattern. The origin of this pattern is the top-left corner of the ActiveArea rectangle. The values are stored in row-column-sample scan order.

See Chapter 5, "Mapping Raw Values to Linear Reference Values" on page 99 for details of the processing model.

BlackLevelDeltaH

Tag 50715 (C61B.H)

Type SRATIONAL

Count ActiveArea width

Value See below

Default All zeros

Usage Raw IFD

Description

If the zero light encoding level is a function of the image column, this tag specifies the difference between the zero light encoding level for each column and the baseline zero light encoding level.

If SamplesPerPixel is not equal to one, this single table applies to all the samples for each pixel.

See Chapter 5, "Mapping Raw Values to Linear Reference Values" on page 99 for details of the processing model.

BlackLevelDeltaV

Tag 50716 (C61C.H)

Type SRATIONAL

Count ActiveArea length

Value See below

Default All zeros

Usage Raw IFD

Description

If the zero light encoding level is a function of the image column, this tag specifies the difference between the zero light encoding level for each row and the baseline zero light encoding level.

If SamplesPerPixel is not equal to one, this single table applies to all the samples for each pixel.

See Chapter 5, "Mapping Raw Values to Linear Reference Values" on page 99 for details of the processing model.

WhiteLevel

Tag 50717 (C61D.H)

Type SHORT or LONG

Count SamplesPerPixel

Value See below

Default $2^{BitsPerSample} - 1$

Usage Raw IFD

Description

This tag specifies the fully saturated encoding level for the raw sample values. Saturation is caused either by the sensor itself becoming highly non-linear in response, or by the camera's analog to digital converter clipping.

The default value for this tag is $2^{BitsPerSample} - 1$ for unsigned integer images, and 1.0 for floating point images.

See Chapter 5, "Mapping Raw Values to Linear Reference Values" on page 99 for details of the processing model.

DefaultScale

Tag 50718 (C61E.H)

Type RATIONAL

Count 2

Value DefaultScaleH, DefaultScaleV

Default 1.0, 1.0

Usage Raw IFD

Description

DefaultScale is required for cameras with non-square pixels. It specifies the default scale factors for each direction to convert the image to square pixels. Typically these factors are selected to approximately preserve total pixel count.

For CFA images that use CFALayout equal to 2, 3, 4, or 5, such as the Fujifilm SuperCCD, these two values should usually differ by a factor of 2.0.

BestQualityScale

Tag 50780 (C65C.H)

Type RATIONAL

Count 1

Value See below

Default 1.0

Usage Raw IFD

Description

For some cameras, the best possible image quality is not achieved by preserving the total pixel count during conversion. For example, Fujifilm SuperCCD images have maximum detail when their total pixel count is doubled.

This tag specifies the amount by which the values of the DefaultScale tag need to be multiplied to achieve the best quality image size.

DefaultCropOrigin

Tag 50719 (C61F.H)

Type SHORT or LONG or RATIONAL

Count 2

Value DefaultCropOriginH, DefaultCropOriginV

Default 0, 0

Usage Raw IFD

Description

Raw images often store extra pixels around the edges of the final image. These extra pixels help prevent interpolation artifacts near the edges of the final image.

DefaultCropOrigin specifies the origin of the final image area, in raw image coordinates (i.e., before the DefaultScale has been applied), relative to the top-left corner of the ActiveArea rectangle.

DefaultCropSize

Tag 50720 (C620.H)

Type SHORT or LONG or RATIONAL

Count 2

Value DefaultCropSizeH, DefaultCropSizeV

Default ImageWidth, ImageLength

Usage Raw IFD

Description

Raw images often store extra pixels around the edges of the final image. These extra pixels help prevent interpolation artifacts near the edges of the final image.

DefaultCropSize specifies the size of the final image area, in raw image coordinates (i.e., before the DefaultScale has been applied).

CalibrationIlluminant1

Tag 50778 (C65A.H)

Type SHORT

Count 1

Value See below

Default 0 (unknown)

Usage IFD 0 or Camera Profile IFD

Description

The illuminant used for the first set of color calibration tags. The legal values for this tag are the same as the legal values for the LightSource EXIF tag.

Starting with DNG 1.6.0.0, the value may be set to 255 (Other) to indicate a custom illuminant. If set to 255, then the IFD must also include a IlluminantData1 tag to specify the x-y chromaticity or spectral power distribution function for this illuminant.

See Chapter 6, "Mapping Camera Color Space to CIE XYZ Space" on page 100 for details of the color-processing model.

CalibrationIlluminant2

Tag 50779 (C65B.H)

Type SHORT

Count 1

Value See below

Default None

Usage IFD 0 or Camera Profile IFD

Description

The illuminant used for an optional second set of color calibration tags. The legal values for this tag are the same as the legal values for the CalibrationIlluminant1 tag; however, if both are included, neither is allowed to have a value of 0 (unknown).

Starting with DNG 1.6.0.0, the value may be set to 255 (Other) to indicate a custom illuminant. If set to 255, then the IFD must also include a IlluminantData2 tag to specify the x-y chromaticity or spectral power distribution function for this illuminant.

See Chapter 6, "Mapping Camera Color Space to CIE XYZ Space" on page 100 for details of the color-processing model.

ColorMatrix1

Tag 50721 (C621.H)

Type SRATIONAL

Count ColorPlanes * 3

Value See below

Default None

Usage IFD 0 or Camera Profile IFD

Description

ColorMatrix1 defines a transformation matrix that converts XYZ values to reference camera native color space values, under the first calibration illuminant. The matrix values are stored in row scan order.

The ColorMatrix1 tag is required for all non-monochrome DNG files.

See Chapter 6, "Mapping Camera Color Space to CIE XYZ Space" on page 100 for details of the color-processing model.

ColorMatrix2

Tag 50722 (C622.H)

Type SRATIONAL

Count ColorPlanes * 3

Value See below

Default None

Usage IFD 0 or Camera Profile IFD

Description

ColorMatrix2 defines a transformation matrix that converts XYZ values to reference camera native color space values, under the second calibration illuminant. The matrix values are stored in row scan order.

See Chapter 6, "Mapping Camera Color Space to CIE XYZ Space" on page 100 for details of the color-processing model.

CameraCalibration1

Tag 50723 (C623.H)

Type SRATIONAL

Count ColorPlanes * ColorPlanes

Value See below

Default Identity matrix

Usage IFD 0

Description

CameraCalibration1 defines a calibration matrix that transforms reference camera native space values to individual camera native space values under the first calibration illuminant. The matrix is stored in row scan order.

This matrix is stored separately from the matrix specified by the ColorMatrix1 tag to allow raw converters to swap in replacement color matrices based on UniqueCameraModel tag, while still taking advantage of any per-individual camera calibration performed by the camera manufacturer.

See Chapter 6, "Mapping Camera Color Space to CIE XYZ Space" on page 100 for details of the color-processing model.

CameraCalibration2

Tag 50724 (C624.H)

Type SRATIONAL

Count ColorPlanes * ColorPlanes

Value See below

Default Identity matrix

Usage IFD 0

Description

CameraCalibration2 defines a calibration matrix that transforms reference camera native space values to individual camera native space values under the second calibration illuminant. The matrix is stored in row scan order.

This matrix is stored separately from the matrix specified by the ColorMatrix2 tag to allow raw converters to swap in replacement color matrices based on UniqueCameraModel tag, while still taking advantage of any per-individual camera calibration performed by the camera manufacturer.

See Chapter 6, "Mapping Camera Color Space to CIE XYZ Space" on page 100 for details of the color-processing model.

ReductionMatrix1

Tag 50725 (C625.H)

Type SRATIONAL

Count 3 * ColorPlanes

Value See below

Default None

Usage IFD 0 or Camera Profile IFD

Description

ReductionMatrix1 defines a dimensionality reduction matrix for use as the first stage in converting color camera native space values to XYZ values, under the first calibration illuminant. This tag may only be used if ColorPlanes is greater than 3. The matrix is stored in row scan order.

See Chapter 6, "Mapping Camera Color Space to CIE XYZ Space" on page 100 for details of the color-processing model.

ReductionMatrix2

Tag 50726 (C626.H)

Type SRATIONAL

Count 3 * ColorPlanes

Value See below

Default None

Usage IFD 0 or Camera Profile IFD

Description

ReductionMatrix2 defines a dimensionality reduction matrix for use as the first stage in converting color camera native space values to XYZ values, under the second calibration illuminant. This tag may only be used if ColorPlanes is greater than 3. The matrix is stored in row scan order.

See Chapter 6, "Mapping Camera Color Space to CIE XYZ Space" on page 100 for details of the color-processing model.

AnalogBalance

Tag 50727 (C627.H)

Type RATIONAL

Count ColorPlanes

Value See below

Default All 1.0

Usage IFD 0

Description

Normally the stored raw values are not white balanced, since any digital white balancing will reduce the dynamic range of the final image if the user decides to later adjust the white balance; however, if camera hardware is capable of white balancing the color channels before the signal is digitized, it can improve the dynamic range of the final image.

AnalogBalance defines the gain, either analog (recommended) or digital (not recommended) that has been applied the stored raw values.

See Chapter 6, "Mapping Camera Color Space to CIE XYZ Space" on page 100 for details of the color-processing model.

AsShotNeutral

Tag 50728 (C628.H)

Type SHORT or RATIONAL

Count ColorPlanes

Value See below

Default None

Usage IFD 0

Description

AsShotNeutral specifies the selected white balance at time of capture, encoded as the coordinates of a perfectly neutral color in linear reference space values. The inclusion of this tag precludes the inclusion of the AsShotWhiteXY tag.

AsShotWhiteXY

Tag 50729 (C629.H)

Type RATIONAL

Count 2

Value See below

Default None

Usage IFD 0

Description

AsShotWhiteXY specifies the selected white balance at time of capture, encoded as x-y chromaticity coordinates. The inclusion of this tag precludes the inclusion of the AsShotNeutral tag.

BaselineExposure

Tag 50730 (C62A.H)

Type SRATIONAL

Count 1

Value See below

Default 0.0

Usage IFD 0

Description

Camera models vary in the tradeoff they make between highlight headroom and shadow noise. Some leave a significant amount of highlight headroom during a normal exposure. This allows significant negative exposure compensation to be applied during raw conversion, but also means normal exposures will contain more

shadow noise. Other models leave less headroom during normal exposures. This allows for less negative exposure compensation but results in lower shadow noise for normal exposures.

Because of these differences, a raw converter needs to vary the zero point of its exposure compensation control from model to model. BaselineExposure specifies by how much (in EV units) to move the zero point. Positive values result in brighter default results, while negative values result in darker default results.

BaselineNoise

Tag 50731 (C62B.H)

Type RATIONAL

Count 1

Value See below

Default 1.0

Usage IFD 0

Description

BaselineNoise specifies the relative noise level of the camera model at a baseline ISO value of 100, compared to a reference camera model.

Since noise levels tend to vary approximately with the square root of the ISO value, a raw converter can use this value, combined with the current ISO, to estimate the relative noise level of the current image.

It is recommended that cameras use the newer NoiseProfile tag instead.

BaselineSharpness

Tag 50732 (C62C.H)

Type RATIONAL

Count 1

Value See below

Default 1.0

Usage IFD 0 or Enhanced IFD

Description

BaselineSharpness specifies the relative amount of sharpening required for this camera model, compared to a reference camera model. Camera models vary in the strengths of their anti- aliasing filters. Cameras with weak or no filters require less sharpening than cameras with strong anti-aliasing filters.

BayerGreenSplit

Tag 50733 (C62D.H)

Type LONG

Count 1

Value See below

Default 0

Usage Raw IFD

Description

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 (quite large divergence).

LinearResponseLimit

Tag 50734 (C62E.H)

Type RATIONAL

Count 1

Value See below

Default 1.0

Usage IFD 0

Description

Some sensors have an unpredictable non-linearity in their response as they near the upper limit of their encoding range. This non-linearity results in color shifts in the highlight areas of the resulting image unless the raw converter compensates for this effect.

LinearResponseLimit specifies the fraction of the encoding range above which the response may become significantly non-linear.

CameraSerialNumber

Tag 50735 (C62F.H)

Type ASCII

Count String length including null

Value Null-terminated string

Default None

Usage IFD 0

Description

CameraSerialNumber contains the serial number of the camera or camera body that captured the image.

LensInfo

Tag 50736 (C630.H)

Type RATIONAL

Count 4

Value Value 0: Minimum focal length in mm.

Value 1: Maximum focal length in mm.

Value 2: Minimum f-stop (maximum aperture) at minimum focal length.

Value 3: Minimum f-stop (maximum aperture) at maximum focal length.

Default None

Usage IFD 0

Description

LensInfo contains information about the lens that captured the image. If the minimum f-stops are unknown, they should be encoded as 0/0.

ChromaBlurRadius

Tag 50737 (C631.H)

Type RATIONAL

Count 1

Value Chroma blur radius in pixels

Default See below

Usage Raw IFD

Description

ChromaBlurRadius provides a hint to the DNG reader about how much chroma blur should be applied to the image. If this tag is omitted, the reader will use its default amount of chroma blurring.

Normally this tag is only included for non-CFA images, since the amount of chroma blur required for mosaic images is highly dependent on the de-mosaic algorithm, in which case the DNG reader's default value is likely optimized for its particular de-mosaic algorithm.

AntiAliasStrength

Tag 50738 (C632.H)

Type RATIONAL

Count 1

Value Relative strength of the camera’s anti-alias filter

Default 1.0

Usage Raw IFD

Description

AntiAliasStrength provides a hint to the DNG reader about how strong the camera's anti-alias filter is. A value of 0.0 means no anti-alias filter (i.e., the camera is prone to aliasing artifacts with some subjects), while a value of 1.0 means a strong anti-alias filter (i.e., the camera almost never has aliasing artifacts).

Note that this tag overlaps in functionality with the BaselineSharpness tag. The primary difference is the AntiAliasStrength tag is used as a hint to the de-mosaic algorithm, while the BaselineSharpness tag is used as a hint to a sharpening algorithm applied later in the processing pipeline.

ShadowScale

Tag 50739 (C633.H)

Type RATIONAL

Count 1

Value See below

Default 1.0

Usage IFD 0

Description

This tag is used by older versions of Adobe Camera Raw to control the sensitivity of its "Blacks" slider.

DNGPrivateData

Tag 50740 (C634.H)

Type BYTE

Count Length of private data block in bytes

Value See below

Default None

Usage IFD 0

Description

This tag provides a way for camera manufacturers to store private data in the DNG file for use by their own raw converters, and to have that data preserved by programs that edit DNG files.

The private data must follow these rules:

- *The private data must start with a null-terminated ASCII string identifying the data*. The first part of this string must be the manufacturer's name, to avoid conflicts between manufacturers.
- *The private data must be self-contained*. All offsets within the private data must be offsets relative to the start of the private data, and they must not point to bytes outside the private data.
- *The private data must be byte-order independent*. If a DNG file is converted from a big- endian file to a little-endian file, the data must remain valid.

MakerNoteSafety

Tag 50741 (C635.H)

Type SHORT

Count 1

Value 0 (unsafe) or 1 (safe)

Default 0

Usage IFD 0

Description

MakerNoteSafety lets the DNG reader know whether the EXIF MakerNote tag is safe to preserve along with the rest of the EXIF data. File browsers and other image management software processing an image with a preserved MakerNote should be aware that any thumbnail image embedded in the MakerNote may be stale and may not reflect the current state of the full-size image.

A MakerNote is safe to preserve if it follows these rules:

- *The MakerNote data must be self-contained*. All offsets within the MakerNote must be offsets relative to the start of the MakerNote, and they must not point to bytes outside the MakerNote.
- *The MakerNote data must be byte-order independent*. Moving the data to a file with a different byte order must not invalidate it.

RawDataUniqueID

Tag 50781 (C65D.H)

Type BYTE

Count 16

Value See below

Default Optional

Usage IFD 0

Description

This tag contains a 16-byte unique identifier for the raw image data in the DNG file. DNG readers can use this tag to recognize a particular raw image, even if the file's name or the metadata contained in the file has been changed.

If a DNG writer creates such an identifier, it should do so using an algorithm that will ensure that it is very unlikely two different images will end up having the same identifier.

OriginalRawFileName

Tag 50827 (C68B.H)

Type ASCII or BYTE

Count Byte count including null

Value Null-terminated UTF-8 encoded Unicode string

Default Optional

Usage IFD 0

Description

If the DNG file was converted from a non-DNG raw file, then this tag contains the file name of that original raw file.

OriginalRawFileData

Tag 50828 (C68C.H)

Type UNDEFINED

Count Byte count of embedded data

Value See below

Default Optional

Usage IFD 0

Description

If the DNG file was converted from a non-DNG raw file, then this tag contains the compressed contents of that original raw file.

The contents of this tag always use the big-endian byte order.

The tag contains a sequence of data blocks. Future versions of the DNG specification may define additional data blocks, so DNG readers should ignore extra bytes when parsing this tag. DNG readers should also detect the case where data blocks are missing from the end of the sequence and should assume a default value for all the missing blocks.

There are no padding or alignment bytes between data blocks. The sequence of data blocks is:

1. Compressed data fork of original raw file.
2. Compressed macOS resource fork of original raw file.
3. macOS file type (4 bytes) of original raw file.
4. macOS file creator (4 bytes) of original raw file.
5. Compressed data fork of sidecar ".THM" file.
6. Compressed macOS resource fork of sidecar ".THM" file.
7. macOS file type (4 bytes) of sidecar ".THM" file.
8. macOS file creator (4 bytes) of sidecar ".THM" file.

If the macOS file types or creator codes are unknown, zero is stored.

If the macOS resource forks do not exist, they should be encoded as zero-length forks. Each fork (data or macOS resource) is compressed and encoded as:

> ForkLength = first four bytes. This is the uncompressed length of this fork. If this value is zero, then no more data is stored for this fork.

From ForkLength, compute the number of 64K compression blocks used for this data (the last block is usually smaller than 64K):

> ForkBlocks = Floor ((ForkLength + 65535) / 65536)

The next (ForkBlocks + 1) 4-byte values are an index into the compressed data. The first ForkBlock values are offsets from the start of the data for this fork to the start of the compressed data for the corresponding compression block. The last value is an offset from the start of the data for this fork to the end of the data for this fork.

Following this index is the ZIP compressed data for each 64K compression block.

ActiveArea

Tag 50829 (C68D.H)

Type SHORT or LONG

Count 4

Value See below

Default 0, 0, ImageLength, ImageWidth

Usage Raw IFD

Description

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

MaskedAreas

Tag 50830 (C68E.H)

Type SHORT or LONG

Count 4 * number of rectangles

Value See below

Default None

Usage Raw IFD

Description

This tag contains a list of non-overlapping rectangle coordinates of fully masked pixels, which 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 estimate and store the black encoding level using the black level DNG tags. Support for the MaskedAreas tag is not required of DNG readers.

AsShotICCProfile

Tag 50831 (C68F.H)

Type UNDEFINED

Count Length of ICC profile in bytes

Value See below

Default Optional

Usage IFD 0

Description

This tag contains an ICC profile that, in conjunction with the AsShotPreProfileMatrix tag, provides the camera manufacturer with a way to specify a default color rendering from camera color space coordinates (linear reference values) into the ICC profile connection space.

The ICC profile connection space is an output referred colorimetric space, whereas the other color calibration tags in DNG specify a conversion into a scene referred colorimetric space. This means that the rendering in this profile should include any desired tone and gamut mapping needed to convert between scene referred values and output referred values.

DNG readers that have their own tone and gamut mapping controls (such as Adobe Camera Raw) will probably ignore this tag pair.

AsShotPreProfileMatrix

Tag 50832 (C690.H)

Type SRATIONAL

Count 3 * ColorPlanes or

ColorPlanes * ColorPlanes

Value See below

Default Identity matrix

Usage IFD 0

Description

This tag is used in conjunction with the AsShotICCProfile tag. It specifies a matrix that should be applied to the camera color space coordinates before processing the values through the ICC profile specified in the AsShotICCProfile tag.

The matrix is stored in the row scan order.

If ColorPlanes is greater than three, then this matrix can (but is not required to) reduce the dimensionality of the color data down to three components, in which case the AsShotICCProfile should have three rather than ColorPlanes input components.

CurrentICCProfile

Tag 50833 (C691.H)

Type UNDEFINED

Count Length of ICC profile in bytes

Value See below

Default Optional

Usage IFD 0

Description

This tag is used in conjunction with the CurrentPreProfileMatrix tag.

The CurrentICCProfile and CurrentPreProfileMatrix tags have the same purpose and usage as the AsShotICCProfile and AsShotPreProfileMatrix tag pair, except they are for use by raw file editors rather than camera manufacturers.

CurrentPreProfileMatrix

Tag 50834 (C692.H)

Type SRATIONAL

Count 3 * ColorPlanes or

ColorPlanes * ColorPlanes

Value See below

Default Identity matrix

Usage IFD 0

Description

This tag is used in conjunction with the CurrentICCProfile tag.

The CurrentICCProfile and CurrentPreProfileMatrix tags have the same purpose and usage as the AsShotICCProfile and AsShotPreProfileMatrix tag pair, except they are for use by raw file editors rather than camera manufacturers.

Additional Tags for Version 1.2.0.0

The following tags have been added for the 1.2.0.0 version of this specification.

ColorimetricReference

Tag 50879 (C6BF.H)

Type SHORT

Count 1

Value 0, 1, or 2

Default 0

Usage IFD 0

Description

The DNG color model documents a transform between camera colors and CIE XYZ values. This tag describes the colorimetric reference for the CIE XYZ values.

- 0 = The XYZ values are scene-referred.
- 1 = The XYZ values are output-referred, using the ICC profile perceptual dynamic range.
- 2 = The XYZ values are output-referred and may be high dynamic range. XYZ coordinates may be less than 0.0 or greater than 1.0.

Values 1 and 2 allow output-referred data to be stored in DNG files and still processed correctly by DNG readers.

Values 0 and 1 are supported in DNG version 1.2 and later.

Value 2 is supported in DNG version 1.7 and later.

CameraCalibrationSignature

Tag 50931 (C6F3.H)

Type ASCII or BYTE

Count Length of string including null

Value Null-terminated string

Default Empty string

Usage IFD 0

Description

A UTF-8 encoded string associated with the CameraCalibration1 and CameraCalibration2 tags. The CameraCalibration1 and CameraCalibration2 tags should only be used in the DNG color transform if the string stored in the CameraCalibrationSignature tag exactly matches the string stored in the ProfileCalibrationSignature tag for the selected camera profile.

ProfileCalibrationSignature

Tag 50932 (C6F4.H)

Type ASCII or BYTE

Count Length of string including null

Value Null-terminated string

Default Empty string

Usage IFD 0 or Camera Profile IFD

Description

A UTF-8 encoded string associated with the camera profile tags. The CameraCalibration1 and CameraCalibration2 tags should only be used in the DNG color transform if the string stored in the CameraCalibrationSignature tag exactly matches the string stored in the ProfileCalibrationSignature tag for the selected camera profile.

ExtraCameraProfiles

Tag 50933 (C6F5.H)

Type LONG

Count Number of extra camera profiles

Value Offsets to Camera Profile IFDs

Default Empty list

Usage IFD 0

Description

A list of file offsets to extra Camera Profile IFDs. The format of a camera profile begins with a 16-bit byte order mark (MM or II) followed by a 16-bit "magic" number equal to 0x4352 (CR), a 32-bit IFD offset, and then a standard TIFF format IFD. All offsets are relative to the start of the byte order mark. Note that the primary camera profile tags should be stored in IFD 0, and the ExtraCameraProfiles tag should only be used if there is more than one camera profile stored in the DNG file.

AsShotProfileName

Tag 50934 (C6F6.H)

Type ASCII or BYTE

Count Length of string including null

Value Null-terminated string

Default Optional

Usage IFD 0

Description

A UTF-8 encoded string containing the name of the "as shot" camera profile, if any.

NoiseReductionApplied

Tag 50935 (C6F7.H)

Type RATIONAL

Count 1

Value See below

Default 0/0

Usage Raw IFD or Enhanced IFD

Description

This tag indicates how much noise reduction has been applied to the raw data on a scale of 0.0 to 1.0. A 0.0 value indicates that no noise reduction has been applied. A 1.0 value indicates that the "ideal" amount of noise reduction has been applied, i.e., that the DNG reader should not apply additional noise reduction by default. A value of 0/0 indicates that this parameter is unknown.

ProfileName

Tag 50936 (C6F8.H)

Type ASCII or BYTE

Count Length of string including null

Value Null-terminated string

Default Optional

Usage IFD 0 or Camera Profile IFD

Description

A UTF-8 encoded string containing the name of the camera profile. This tag is optional if there is only a single camera profile stored in the file but is required for all camera profiles if there is more than one camera profile stored in the file.

ProfileHueSatMapDims

Tag 50937 (C6F9.H)

Type LONG

Count 3

Value HueDivisions, SaturationDivisions, ValueDivisions

Default None

Usage IFD 0 or Camera Profile IFD

Description

This tag specifies the number of input samples in each dimension of the hue/saturation/value mapping tables. The data for these tables are stored in ProfileHueSatMapData1 and ProfileHueSatMapData2 tags. Allowed values include the following:

- HueDivisions >= 1
- SaturationDivisions >= 2
- ValueDivisions >=1

The most common case has ValueDivisions equal to 1, so only hue and saturation are used as inputs to the mapping table.

ProfileHueSatMapData1

Tag 50938 (C6FA.H)

Type FLOAT

Count HueDivisions * SaturationDivisions * ValueDivisions * 3

Value See below

Default None

Usage IFD 0 or Camera Profile IFD

Description

This tag contains the data for the first hue/saturation/value mapping table. Each entry of the table contains three 32-bit IEEE floating-point values. The first entry is hue shift in degrees; the second entry is saturation scale factor; and the third entry is a value scale factor. The table entries are stored in the tag in nested loop order, with the value divisions in the outer loop, the hue divisions in the middle loop, and the saturation divisions in the inner loop. All zero input saturation entries are required to have a value scale factor of 1.0. The hue/saturation/value table application is described in detail in Chapter 6.

ProfileHueSatMapData2

Tag 50939 (C6FB.H)

Type FLOAT

Count HueDivisions * SaturationDivisions * ValueDivisions * 3

Value See below

Default None

Usage IFD 0 or Camera Profile IFD

Description

This tag contains the data for the second hue/saturation/value mapping table. Each entry of the table contains three 32-bit IEEE floating-point values. The first entry is hue shift in degrees; the second entry is saturation scale factor; and the third entry is a value scale factor. The table entries are stored in the tag in

nested loop order, with the value divisions in the outer loop, the hue divisions in the middle loop, and the saturation divisions in the inner loop. All zero input saturation entries are required to have a value scale factor of 1.0. The hue/saturation/value table application is described in detail in Chapter 6.

ProfileToneCurve

Tag 50940 (C6FC.H)

Type FLOAT

Count Samples * 2

Value See below

Default None

Usage IFD 0 or Camera Profile IFD

Description

This tag contains a default tone curve that can be applied while processing the image as a starting point for user adjustments. The curve is specified as a list of 32-bit IEEE floating- point value pairs in linear gamma. Each sample is a pair (x,y) where x is the input value in the range of 0.0 to 1.0 and y is the output value in the range of 0.0 to 1.0. Samples must be stored in ascending order of x coordinates, and each sample must have an x coordinate that is greater than the previous sample (strictly increasing).

DNG readers should interpolate the curve using a cubic spline.

For Standard Dynamic Range camera profiles, the first sample is required to be (0.0, 0.0), and the last sample is required to be (1.0, 1.0).

For High Dynamic Range camera profiles, the first sample is required to be (0.0, 0.0). The curve is applied in an encoded space (instead of linear gamma) to accommodate overrange pixel values. See the description of ProfileDynamicRange for details. Readers should process pixel values beyond (greater than) the curve's final point by using slope extension; that is, the affine function that passes through the curve's final point and matches the curve's slope at that point.

ProfileEmbedPolicy

Tag 50941 (C6FD.H)

Type LONG

Count 1

Value See below

Default 0

Usage IFD 0 or Camera Profile IFD

Description

This tag contains information about the usage rules for the associated camera profile. The valid values and meanings are:

- 0 = “allow copying”. The camera profile can be used to process, or be embedded in, any DNG file. It can be copied from DNG files to other DNG files or copied from DNG files and stored on the user’s system for use in processing or embedding in any DNG file. The camera profile may not be used to process non-DNG files.
- 1 = “embed if used”. This value applies the same rules as “allow copying”, except it does not allow copying the camera profile from a DNG file for use in processing any image other than the image in which it is embedded, unless the profile is already stored on the user’s system.
- 2 = “embed never”. This value only applies to profiles stored on a user’s system but not already embedded in DNG files. These stored profiles can be used to process images but cannot be embedded in files. If a camera profile is already embedded in a DNG file, then this value has the same restrictions as “embed if used”.
- 3 = “no restrictions”. The camera profile creator has not placed any restrictions on the use of the camera profile.

ProfileCopyright

Tag 50942 (C6FE.H)

Type ASCII or BYTE

Count Length of string including null

Value Null-terminated string

Default Optional

Usage IFD 0 or Camera Profile IFD

Description

A UTF-8 encoded string containing the copyright information for the camera profile. This string always should be preserved along with the other camera profile tags.

ForwardMatrix1

Tag 50964 (C714.H)

Type SRATIONAL

Count 3 * ColorPlanes

Value See below

Default Optional

Usage IFD 0 or Camera Profile IFD

Description

This tag defines a matrix that maps white balanced camera colors to XYZ D50 colors. Application is described in detail in Chapter 6.

ForwardMatrix2

Tag 50965 (C715.H)

Type SRATIONAL

Count 3 * ColorPlanes

Value See below

Default Optional

Usage IFD 0 or Camera Profile IFD

Description

This tag defines a matrix that maps white balanced camera colors to XYZ D50 colors. Application is described in detail in Chapter 6.

PreviewApplicationName

Tag 50966 (C716.H)

Type ASCII or BYTE

Count Length of string including null

Value Null-terminated string

Default Optional

Usage Preview IFD

Description

A UTF-8 encoded string containing the name of the application that created the preview stored in the IFD.

PreviewApplicationVersion

Tag 50967 (C717.H)

Type ASCII or BYTE

Count Length of string including null

Value Null-terminated string

Default Optional

Usage Preview IFD

Description

A UTF-8 encoded string containing the version number of the application that created the preview stored in the IFD.

PreviewSettingsName

Tag 50968 (C718.H)

Type ASCII or BYTE

Count Length of string including null

Value Null-terminated string

Default Optional

Usage Preview IFD

Description

A UTF-8 encoded string containing the name of the conversion settings (for example, snapshot name) used for the preview stored in the IFD.

PreviewSettingsDigest

Tag 50969 (C719.H)

Type BYTE

Count 16

Value See below

Default Optional

Usage Preview IFD

Description

A unique ID of the conversion settings (for example, MD5 digest) used to render the preview stored in the IFD.

PreviewColorSpace

Tag 50970 (C71A.H)

Type LONG

Count 1

Value See below

Default See below

Usage Preview IFD

Description

This tag specifies the color space in which the rendered preview in this IFD is stored. The valid values include:

- 0 = Unknown
- 1 = Gray Gamma 2.2

- 2 = sRGB
- 3 = Adobe RGB
- 4 = ProPhoto RGB

The default value for this tag is sRGB for color previews and Gray Gamma 2.2 for monochrome previews.

PreviewDateTime

Tag 50971 (C71B.H)

Type ASCII

Count Length of string including null

Value Null-terminated string

Default Optional

Usage Preview IFD

Description

This tag is an ASCII string containing the name of the date/time at which the preview stored in the IFD was rendered. The date/time is encoded using ISO 8601 format.

RawImageDigest

Tag 50972 (C71C.H)

Type BYTE

Count 16

Value See below

Default Optional

Usage IFD 0

Description

This tag is an MD5 digest of the raw image data. All pixels in the image are processed in row- scan order. Each pixel is zero padded to 16 or 32 bits deep (16-bit for data less than or equal to 16 bits deep, 32-bit otherwise). The data for each pixel is processed in little-endian byte order.

OriginalRawFileDigest

Tag 50973 (C71D.H)

Type BYTE

Count 16

Value See below

Default Optional

Usage IFD 0

Description

This tag is an MD5 digest of the data stored in the OriginalRawFileData tag.

SubTileBlockSize

Tag 50974 (C71E.H)

Type SHORT or LONG

Count 2

Value SubTileBlockRows, SubTileBlockCols

Default 1, 1

Usage Raw IFD

Description

Normally, the pixels within a tile are stored in simple row-scan order. This tag specifies that the pixels within a tile should be grouped first into rectangular blocks of the specified size. These blocks are stored in row-scan order. Within each block, the pixels are stored in row-scan order. The use of a non-default value for this tag requires setting the DNGBackwardVersion tag to at least 1.2.0.0.

RowInterleaveFactor

Tag 50975 (C71F.H)

Type SHORT or LONG

Count 1

Value RowInterleaveFactor

Default 1

Usage Raw IFD

Description

This tag specifies that rows of the image are stored in interleaved order. The value of the tag specifies the number of interleaved fields. The use of a non-default value for this tag requires setting the DNGBackwardVersion tag to at least 1.2.0.0.

ProfileLookTableDims

Tag 50981 (C725.H)

Type LONG

Count 3

Value HueDivisions, SaturationDivisions, ValueDivisions

Default none

Usage IFD 0 or Camera Profile IFD

Description

This tag specifies the number of input samples in each dimension of a default "look" table. The data for this table is stored in the ProfileLookTableData tag. Allowed values include:

HueDivisions >= 1

SaturationDivisions >= 2

ValueDivisions >= 1

ProfileLookTableData

Tag 50982 (C726.H)

Type FLOAT

Count HueDivisions * SaturationDivisions * ValueDivisions * 3

Value See below

Default none

Usage IFD 0 or Camera Profile IFD

Description

This tag contains a default "look" table that can be applied while processing the image as a starting point for user adjustment. This table uses the same format as the tables stored in the ProfileHueSatMapData1 and ProfileHueSatMapData2 tags and is applied in the same color space. However, it should be applied later in the processing pipe, after any exposure compensation and/or fill light stages, but before any tone curve stage.

Each entry of the table contains three 32-bit IEEE floating-point values. The first entry is hue shift in degrees, the second entry is a saturation scale factor, and the third entry is a value scale factor.

The table entries are stored in the tag in nested loop order, with the value divisions in the outer loop, the hue divisions in the middle loop, and the saturation divisions in the inner loop.

All zero input saturation entries are required to have a value scale factor of 1.0.

Note that High Dynamic Range profiles affect the application of this tag. See ProfileDynamicRange and the section on Applying the Hue/Value/Saturation Mapping Table for details.

Additional Tags for Version 1.3.0.0

The following tags have been added for the 1.3.0.0 version of this specification.

OpcodeList1

Tag 51008 (C740.H)

Type UNDEFINED

Count Variable

Value Opcode List

Default Empty List

Usage Raw IFD

Description

Specifies the list of opcodes that should be applied to the raw image, as read directly from the file. The format and processing details of an opcode list are described in Chapter 7, "Opcode List Processing."

OpcodeList2

Tag 51009 (C741.H)

Type UNDEFINED

Count Variable

Value Opcode List

Default Empty List

Usage Raw IFD

Description

Specifies the list of opcodes that should be applied to the raw image, just after it has been mapped to linear reference values. The format and processing details of an opcode list are described in Chapter 7, "Opcode List Processing."

OpcodeList3

Tag 51022 (C74E.H)

Type UNDEFINED

Count Variable

Value Opcode List

Default Empty List

Usage Raw IFD

Description

Specifies the list of opcodes that should be applied to the raw image, just after it has been demosaiced. The format and processing details of an opcode list are described in Chapter 7, "Opcode List Processing."

NoiseProfile

Tag 51041 (C761.H)

Type DOUBLE

Count 2 or 2 * ColorPlanes

Value See below

Default Values are estimated from BaselineNoise tag (see below)

Usage Raw IFD or Enhanced IFD

Description

NoiseProfile describes the amount of noise in a raw image. Specifically, this tag models the amount of signal-dependent photon (shot) noise and signal-independent sensor readout noise, two common sources of noise in raw images. The model assumes that the noise is white and spatially independent, ignoring fixed pattern effects and other sources of noise (e.g., pixel response non-uniformity, spatially-dependent thermal effects, etc.).

This tag is intended to be used to describe the amount of noise present in unprocessed raw image data. When noise reduction has already been applied to the raw data (i.e., NoiseReductionApplied > 0), this tag may be used to estimate the white component of the residual noise.

For the purposes of this tag, noise is defined as the standard deviation of a random variable x, where x represents a recorded linear signal in the range $x \in [0,1]$. The two-parameter noise model is:

$$N_i(x) = \sqrt{S_i x + O_i}$$

For parameters (S_i, O_i), where S_i is a scale term that models the amount of sensor amplification, and O_i is an offset term that models the amount of sensor readout noise. A more detailed explanation of this model is given below.

The data elements for this tag are the n sets of noise model parameters:

$$S_1, O_1, S_2, O_2, \ldots, S_n, O_n$$

Note that n must be 1 (i.e., tag count is 2) or equal to the number of color planes in the image (i.e., tag count is 2 $\times$ ColorPlanes). When n = 1, the two specified parameters (S_1, O_1) define the same noise model for all image planes. When n is equal to the number of image planes, the parameters (S_i, O_i) define the noise model for the ith image plane, e.g., (S_1, O_1) correspond to the first image plane, (S_2, O_2) correspond to the second image plane, etc. The order of the parameters follows the plane order specified by the CFAPlaneColor tag.

Each S_i term must be positive $(S_i > 0)$, and each O_i term must be non-negative $(O_i \geq 0)$.

A BaselineNoise tag value of 1.0 at ISO 100 corresponds approximately to NoiseProfile parameter values of $S_i = 2 \times 10^{-5}$ and $O_i = 4.5 \times 10^{-7}$ (e.g., standard deviation of approximately 0.00201 when $x = 0.18$); these values may be used to estimate absolute noise levels in an image when the NoiseProfile tag is missing. When both tags are present, however, DNG readers should prefer using the NoiseProfile data, since it describes noise levels more precisely than BaselineNoise.

A more detailed description of the noise model is given below. This tag models two common sources of noise:

1. Photon (shot) noise p, which has a white Poisson distribution, and
2. Electronic readout noise r, which is present even in the absence of light and is assumed to have an approximately white normal (Gaussian) distribution.

Assuming that p and r are independent random variables, the square of the total noise (i.e., the variance) can be expressed as the sum of the squares of the individual sources of noise:

$$N^2 = p^2 + r^2$$

In this expression, the variables N, p, and r are expressed in B-bit recorded digital values, where common values of B include 12, 14, and 16 bits. If $\hat{x}$ is the average signal level expressed in photons, then its variance will also be $\hat{x}$, since a random variable with a Poisson distribution has a variance equal to its mean:

$$\hat{p}^2 = \hat{x}$$

where $\hat{p}$ denotes the photon noise, expressed in photons. The conversion factor between photons $(\hat{x}, \hat{p})$ and B-bit digital values (x, p) is the gain factor g:

$$x = g \cdot \hat{x}$$

$$p = g \cdot \hat{p}$$

Substituting the 2nd, 3rd, and 4th equations into the first equation yields:

$$N^2 = (g \cdot \hat{p})^2 + r^2$$

$$= (g^2 \cdot \hat{x}) + r^2$$

$$= (g \cdot x) + r^2$$

Therefore, the total noise N can be expressed as a two-parameter function of the signal x:

$$N(x) = \sqrt{g \cdot x + r^2}$$

$$= \sqrt{S_i s + O_i}$$

for model parameters

$$S_i = g$$

$$O_i = r^2$$

This tag uses the convention of a normalized noise model, i.e., $N(x)$ is the standard deviation (i.e., noise) of a random variable x, where x represents a recorded linear signal in the range $x \in [0,1]$. The specified parameters (S_i, O_i) must also be appropriately normalized.

Additional Tags for Version 1.4.0.0

The following tags have been added for the 1.4.0.0 version of this specification.

DefaultUserCrop

Tag 51125 (C7B5.H)

Type RATIONAL

Count 4

Value Top, Left, Bottom, Right

Default 0.0, 0.0, 1.0, 1.0

Usage Raw IFD

Description

Specifies a default user crop rectangle in relative coordinates. The values must satisfy:

- $0 \leq \text{Top} < \text{Bottom} \leq 1.0$
- $0 \leq \text{Left} < \text{Right} \leq 1.0$

The default values of (Top = 0, Left = 0, Bottom = 1, Right = 1) correspond exactly to the default crop rectangle (as specified by the DefaultCropOrigin and DefaultCropSize tags).

DefaultBlackRender

Tag 51110 (C7A6.H)

Type LONG

Count 1

Value See below

Default 0

Usage IFD 0 or Camera Profile IFD

Description

This optional tag in a color profile provides a hint to the raw converter regarding how to handle the black point (e.g., flare subtraction) during rendering. The currently defined values are:

- 0 = Auto
- 1 = None

If set to Auto, the raw converter should perform black subtraction during rendering. The amount and method of black subtraction may be automatically determined and may be image-dependent.

If set to None, the raw converter should not perform any black subtraction during rendering. This may be desirable when using color lookup tables (e.g., LookTable) or tone curves in camera profiles to perform a fixed, consistent level of black subtraction.

BaselineExposureOffset

Tag 51109 (C7A5.H)

Type RATIONAL

Count 1

Value See below

Default 0.0

Usage IFD 0 or Camera Profile IFD

Description

Provides a way for color profiles to increase or decrease exposure during raw conversion.

BaselineExposureOffset specifies the amount (in EV units) to add to the BaselineExposure tag during image rendering. For example, if the BaselineExposure value for a given camera model is +0.3, and the BaselineExposureOffset value for a given camera profile used to render an image for that camera model is -0.7, then the actual default exposure value used during rendering will be +0.3 - 0.7 = -0.4.

ProfileLookTableEncoding

Tag 51108 (C7A4.H)

Type LONG

Count 1

Value See below

Default 0

Usage IFD 0 or Camera Profile IFD

Description

Provides a way for color profiles to specify how indexing into a 3D LookTable is performed during raw conversion. This tag is not applicable to a 2.5D LookTable (i.e., where the Value dimension is 1).

The currently defined values are:

- 0 = Linear encoding
- 1 = sRGB encoding

If set to 0 (Linear encoding), then the method used to apply a 3D LookTable is as follows:

1. Convert linear ProPhoto RGB values to HSV.
2. Use the HSV coordinates to index into the color table.
3. Apply color table result to the original HSV values.
4. Convert modified HSV values back to linear ProPhoto RGB.

If set to 1 (sRGB encoding), then the method used to apply a 3D LookTable is as follows:

1. Convert linear ProPhoto RGB values to HSV.
2. Encode V coordinate using the sRGB encoding curve.
3. Use the encoded HSV coordinates to index into the color table.
4. Apply color table result to the encoded values from step 3.
5. Decode V coordinate using the sRGB decoding curve (inverse of step 2).
6. Convert HSV values back to linear ProPhoto RGB (inverse of step 1).

The second method (sRGB encoding) may be desirable to provide additional table precision to dark (shadow) image values.

Note that High Dynamic Range profiles affect the application of this tag. See ProfileDynamicRange and the section on Applying the Hue/Value/Saturation Mapping Table for details on applying the encoding function (prior to converting RGB to HSV) and applying the decoding function (after converting the HSV values back to RGB).

ProfileHueSatMapEncoding

Tag 51107 (C7A3.H)

Type LONG

Count 1

Value See below

Default 0

Usage IFD 0 or Camera Profile IFD

Description

Provides a way for color profiles to specify how indexing into a 3D HueSatMap is performed during raw conversion. This tag is not applicable to a 2.5D HueSatMap (i.e., where the Value dimension is 1).

The currently defined values are:

- 0 = Linear encoding
- 1 = sRGB encoding

If set to 0 (Linear encoding), then the method used to apply a 3D HueSatMap is as follows:

1. Convert linear ProPhoto RGB values to HSV.
2. Use the HSV coordinates to index into the color table.
3. Apply color table result to the original HSV values.
4. Convert modified HSV values back to linear ProPhoto RGB.

If set to 1 (sRGB encoding), then the method used to apply a 3D HueSatMap is as follows:

1. Convert linear ProPhoto RGB values to HSV.
2. Encode V coordinate using the sRGB encoding curve.
3. Use the encoded HSV coordinates to index into the color table.
4. Apply color table result to the encoded values from step 3.

5. Decode V coordinate using the sRGB decoding curve (inverse of step 2).
6. Convert HSV values back to linear ProPhoto RGB (inverse of step 1).

The second method (sRGB encoding) may be desirable to provide additional table precision to dark (shadow) image values.

Note that High Dynamic Range profiles affect the application of this tag. See ProfileDynamicRange and the section on Applying the Hue/Value/Saturation Mapping Table for details on applying the encoding function (prior to converting RGB to HSV) and applying the decoding function (after converting the HSV values back to RGB).

OriginalDefaultFinalSize

Tag 51089 (C791.H)

Type SHORT or LONG

Count 2

Value width, length

Default See below

Usage IFD

Description

If this file is a proxy for a larger original DNG file, this tag specifics the default final size of the larger original file from which this proxy was generated.

The default value for this tag is default final size of the current DNG file, which is DefaultCropSize * DefaultScale.

OriginalBestQualityFinalSize

Tag 51090 (C792.H)

Type SHORT or LONG

Count 2

Value width, length

Default See below

Usage IFD

Description

If this file is a proxy for a larger original DNG file, this tag specifics the best quality final size of the larger original file from which this proxy was generated.

The default value for this tag is the OriginalDefaultFinalSize, if specified. Otherwise the default value for this tag is the best quality size of the current DNG file, which is DefaultCropSize * DefaultScale * BestQualityScale.

OriginalDefaultCropSize

Tag 51091 (C793.H)

Type SHORT or LONG or RATIONAL

Count 2

Value width, length

Default See below

Usage IFD 0

Description

If this file is a proxy for a larger original DNG file, this tag specifics the DefaultCropSize of the larger original file from which this proxy was generated.

The default value for this tag is the OriginalDefaultFinalSize, if specified. Otherwise, the default value for this tag is the DefaultCropSize of the current DNG file.

NewRawImageDigest

Tag 51111 (C7A7.H)

Type BYTE

Count 16

Value See below

Default Optional

Usage IFD 0

Description

This tag is a modified MD5 digest of the raw image data. It has been updated from the algorithm used to compute the RawImageDigest tag be more multi-processor friendly, and to support lossy compression algorithms. The details of the algorithm used to compute this tag are documented in the Adobe DNG SDK source code.

RawToPreviewGain

Tag 51112 (C7A8.H)

Type DOUBLE

Count 1

Value See below

Default 1.0

Usage Preview IFD

Description

The gain (what number the sample values are multiplied by) between the main raw IFD and the preview IFD containing this tag.

Additional Tags for Version 1.5.0.0

The following tags have been added for the 1.5.0.0 version of this specification.

DepthFormat

Tag 51177 (C7E9.H)

Type SHORT

Count 1

Value See below

Default 0

Usage IFD 0

Description

Specifies the encoding of any depth data in the file. Valid values are:

- 0 - Unknown. The exact mapping of distance values to depth map is unknown, other than nearer distances are closer to zero, and farther distances are closer to the maximum value.
- 1 - Linear. The depth values vary linearly, with zero representing the "near" distance (see the DepthNear tag) and the maximum value representing the "far" distance (see the DepthFar tag).
- 2 - Inverse. The depth values are stored inverse linearly, with zero representing the "near" distance (see the DepthNear tag) and the maximum value representing the "far" distance (see the DepthFar tag).

DepthNear

Tag 51178 (C7EA.H)

Type RATIONAL

Count 1

Value See below

Default 0/0

Usage IFD 0

Description

Specifies distance from the camera represented by the zero value in the depth map. 0/0 means unknown.

DepthFar

Tag 51179 (C7EB.H)

Type RATIONAL

Count 1

Value See below

Default 0/0

Usage IFD 0

Description

Specifies distance from the camera represented by the maximum value in the depth map. 0/0 means unknown. 1/0 means infinity, which is valid for unknown and inverse depth formats.

DepthUnits

Tag 51180 (C7EC.H)

Type SHORT

Count 1

Value See below

Default 0

Usage IFD 0

Description

Specifies the measurement units for the DepthNear and DepthFar tags. Valid values are:

- 0 - Unknown
- 1 - Meters

DepthMeasureType

Tag 51181 (C7ED.H)

Type SHORT

Count 1

Value See below

Default 0

Usage IFD 0

Description

Specifies the measurement geometry for the depth map. Valid values are:

- 0 - Unknown. The measurement geometry is unknown.
- 1 - Optical Axis. Depth is measured along the optical axis.
- 2 - Optical Ray. Depth is measured along the optical ray passing through each pixel.

EnhanceParams

Tag	51182 (C7EE.H)
Type	ASCII
Count	String length including null
Value	Null-terminated string
Default	None
Usage	Enhanced IFD

Description

A string the documents that documents how the enhanced image data was processed. This string is required to be non-null and should begin with the company name that wrote the enhancement algorithms. The remainder of the string is proprietary to that company.

Additional Tags for Version 1.6.0.0

The following tags have been added for the 1.6.0.0 version of this specification.

ProfileGainTableMap

Tag 52525 (CD2D.H)

Type UNDEFINED

Count Byte count of data; see below

Value See below

Default Optional

Usage Raw IFD

Description

This tag contains spatially varying gain tables that can be applied while processing the image as a starting point for user adjustments.

This tag consists of the following parameters, in order:

MapPointsV - LONG

MapPointsH - LONG

MapSpacingV - DOUBLE

MapSpacingH - DOUBLE

MapOriginV - DOUBLE

MapOriginH - DOUBLE

MapPointsN - LONG

MapInputWeights - FLOAT × 5 (e.g., 1/3, 1/3, 1/3, 0, 0)

For Each MapPointsV

For Each MapPointsH

For Each MapPointsN

Gain value - FLOAT

The tag count is the sum of the byte count of all the parameters listed above. This value is 64 + (4 * MapPointsV * MapPointsH * MapPointsN). For example, if MapPointsV = 6, MapPointsH = 8, and MapPointsN = 256, then the total tag size is 64 + (4 * 6 * 8 * 256) = 49216 bytes.

The gain table map is a subsampled 2D matrix of 32-bit floating-point gain tables. MapPointsV is the number of tables in vertical direction. MapPointsH is the number of tables in the horizontal direction.

The gain table map is not required to cover the entire image being modified. Inside the gain table map bounds, the tables are interpolated using bi-linear interpolation. Outside the gain map bounds, values are replicated from the edges of the gain table map.

The origin of the gain table map relative to the image being modified is specified by MapOriginV (vertical direction) and MapOriginH (horizontal direction), which are in relative coordinates, where 1.0 is equal to the height or width of the image being modified, respectively. The origin values can be negative; this means that the gain table map may extend outside the image being modified.

The spacing between gain map points is specified by MapSpacingV (vertical direction) and MapSpacingH (horizontal direction). Again these are in relative coordinates, where 1.0 is equal to the height or width of the image being modified, respectively.

The gain values must be non-negative.

At each point in the image, up to four gain tables are interpolated to determine the gain table to apply at that location. Given the resulting interpolated table, the following method is used to produce an output RGB color given the input RGB color at that point in the image:

1. The "table input value" is a scalar value calculated using the following equation:

   table input value = clamp ((R, G, B, min (R,G,B), max(R,G,B)) · MapInputWeights, 0, 1)

2. Next, the "gain value" is calculated by using "table input value" to perform a linearly interpolated lookup of the gain table.
3. The above gain is multiplied by the input RGB camera values to produce new RGB values. The resulting RGB values may be overrange (greater than 1). DNG readers are not required to clip the values to the nominal [0,1] range; readers are encouraged to preserve the overrange values for potential use in downstream tone adjustments.

ProfileGainTableMap should be applied by the DNG reader in the Reference Input Medium Metric (RIMM) color space. RIMM RGB values are obtained by mapping the native camera RGB values to CIE XYZ (see Chapter 6) and then to RIMM.

ProfileGainTableMap should be applied after the BaselineExposure adjustment; the MapInputWeights values assume that BaselineExposure adjustment has already been applied.

ProfileGainTableMap should be applied after all opcodes have been applied, including all "warp" opcodes that may change the shape of the rendered image. That is, the ProfileGainTableMap x-y grid of gain tables is specified post-warp.

Details on sampling conventions:

- The horizontal and vertical positions of the gain table map should be sampled using pixel-centered (half-pixel) conventions.
- The area of the image to be modified is specified by the ActiveArea tag. For example, if the ActiveArea tag is T=10, L=8, B=3010, R=4008, then the top-left pixel (x,y) = (8,10) corresponds to map sample position (0.5 / 4000, 0.5 / 3000), and the bottom-right pixel (x,y) = (4007, 3009) corresponds to map sample position (3999.5 / 4000, 2999.5 / 3000). Note the half-pixel offset, per the previous item.

- The table input value should be used to index directly into the gain table. That is, multiply the table input value by MapPointsN to compute the floating-point table index.

Additional notes:

- The usage of MapPointsV, MapPointsH, MapOriginV, MapOriginH, MapSpacingV, MapSpacingH is consistent with the definition of these fields from the GainMap opcode (see Chapter 7).
- This tag may be used in combination with the DNG ProfileToneCurve (tag 50940). If used together, the ProfileGainTableMap should be applied first while the image data is in a linear color space.
- The approximate size of this tag given 6 x 8 tables with 256 points is 48 Kb.

SemanticName

Tag 52526 (CD2E.H)

Type ASCII

Count String length including null

Value Null-terminated string

Default Required tag

Usage Semantic Mask IFD

Description

A string that identifies the semantic mask. If a DNG file contains multiple semantic masks, it is strongly recommended that the SemanticName strings in the masks be unique.

SemanticInstanceID

Tag 52528 (CD30.H)

Type ASCII

Count String length including null

Value Null-terminated string

Default Optional

Usage Semantic Mask IFD

Description

A string that identifies a specific instance in a semantic mask.

This tag can be used to relate the content of multiple semantic masks within a single DNG file. For example, suppose the main image in a DNG file is a picture of two persons. The file contains four semantic masks:

1. A mask representing the skin of Person A. SemanticName is set to "skin_mask" and SemanticInstanceID is set to "person_a"
2. A mask representing the hair of Person A. SemanticName is set to "hair_mask" and SemanticInstanceID is set to "person_a"

3. A mask representing the skin of Person B. SemanticName is set to "skin_mask" and SemanticInstanceID is set to "person_b"
4. A mask representing the hair of Person B. SemanticName is set to "hair_mask" and SemanticInstanceID is set to "person_b"

In general, this tag cannot be used to relate content between two different DNG files. For example, semantic masks representing the same person may be assigned different string values across different DNG files.

MaskSubArea

Tag 52536 (CD38.H)

Type LONG

Count 4

Value See below

Default 0, 0, MaskWidth, MaskHeight

Usage Semantic Mask IFD

Description

This tag identifies the crop rectangle of this IFD's mask, relative to the main image. For example, a semantic mask that identifies the foreground subject may be cropped to the subject's extent within the image, thereby reducing file size.

The first two values are the origin of the cropped mask. The last two values are the dimensions of the uncropped mask. Together, these four values (plus the dimensions of the cropped mask, which are provided by the ImageWidth and ImageLength tags in the same IFD), completely describe how the cropped mask relates to the main image.

The four 32-bit unsigned integer values are, in order:

- T_crop = top coordinate of the cropped mask
- L_crop = left coordinate of the cropped mask
- W_full = width of full / uncropped mask
- H_full = height of full / uncropped mask

The term "uncropped mask" refers conceptually to the entire mask, which corresponds to the ActiveArea of the main image.

The term "cropped mask" refers to the actual mask stored in this IFD.

If this tag is absent, it is assumed that the mask covers the entire ActiveArea.

Pixels outside the cropped mask are assumed to have value 0.

If either of the following is true, then the tag is invalid and should be ignored:

- T_crop + ImageWidth > H_full
- L_crop + ImageLength > W_full

Example

The ActiveArea of a DNG has (top, left, bottom, right) coordinate values of (top, left, bottom, right) = (0, 0, 3024, 4032).

The DNG contains a Semantic Mask stored at reduced resolution. The "uncropped mask area" is (0, 0, 800, 1200).

Suppose this Semantic Mask has non-zero pixels for only a small 200x200 sub-rectangle: (100, 80, 300, 280). Thus, it is not necessary to store an entire 1200 x 800 mask image. Instead, it suffices to record a Semantic Mask IFD with ImageWidth = 200, ImageLength = 200, and a MaskSubArea tag with the following values:

Field	Value
T_crop	100
L_crop	80
W_full	1200
H_full	800

If a DNG reader needs to resample the Semantic Mask (e.g., so that it matches the full image resolution), then the correct scaling is the transform that maps (0, 0, 800, 1200) to the ActiveArea (0, 0, 3024, 4032).

RGBTables

Tag	52543 (CD3F.H)
Type	UNDEFINED
Count	See below
Value	See below
Default	Optional
Usage	IFD 0 or Camera Profile IFD

Description

This tag specifies color transforms that can be applied to masked image regions. Color transforms are specified using RGB-to-RGB color lookup tables. These tables are associated with Semantic Masks to limit the color transform to a sub-region of the image. The overall color transform is a linear combination of the color tables, weighted by their corresponding Semantic Masks.

The data format of the tag is:

NumTables – LONG

CompositeMethod – LONG

For each table T (from 1 to NumTables):

data for table T

The format of the data for each table is:

LengthTableSemanticName - SHORT

TableSemanticName - ASCII (not null-terminated)

Divisions - BYTE

PixelType – BYTE

GammaEncoding - BYTE

ColorPrimaries - BYTE

GamutExtension - BYTE

For each red index r

For each green index g

For each blue index b

red value for table entry (r,g,b)

green value for table entry (r,g,b)

blue value for table entry (r,g,b)

Notes on the tag fields

NumTables is the number of tables and must be in the range 1 to 20.

CompositeMethod must be either 0 or 1. A value of 0 means that tables are applied using the Weighted Sum Method. A value of 1 means that tables are applied using the Sequential Method. See below for details on both methods.

LengthTableSemanticName is the byte length of the TableSemanticName tag. A value of 0 represents the special case of a "background table" – see below for details. If multiple tables are specified (NumTables > 1), then at most one of those tables can have a zero value for LengthTableSemanticName.

TableSemanticName is an ASCII string corresponding to one of the SemanticName tags whose mask should be used to apply the table. This tag has length LengthTableSemanticName bytes and is not null-terminated. If LengthTableSemanticName is 0, then this field is omitted. Otherwise, matches between TableSemanticName and SemanticName are made using case-sensitive comparisons; if no match is found, the table is ignored.

Divisions is the linear resolution of the table and must be in the range 2 to 32. The total number of table entries is Divisions x Divisions x Divisions.

PixelType is the pixel type of each table entry and must be 0, 1, or 2. These values have the following meaning:

PixelType	*Meaning*
0	8-bit unsigned integer (0 to 255)
1	16-bit unsigned integer (0 to 65535)
2	32-bit floating-point (0.0 to 1.0)

The **GammaEncoding** tag indicates the input and output gamma encoding of the table. That is, the table should be applied to pixel values that have been encoding using the specified gamma encoding curve; the output pixel values computed by the table will also be encoded using the same curve. This tag must be in the range 0 to 4 and has the following interpretation:

GammaEncoding	*Meaning*
0	Linear
1	sRGB
2	1.8
3	2.2
4	Rec. 2020

The **ColorPrimaries** tag indicates are the input and output RGB primaries of the table. That is, the table should be applied to pixel values that have been mapped to these RGB primaries; the output pixel values computed by the table will also be in the same primaries. This tag must be in the range 0 to 4 and has the following interpretation:

ColorPrimaries	*Meaning*	*Reference White Point*
0	sRGB	D65
1	Adobe RGB	D65
2	Display P3	D65
3	Rec. 2020	D65
4	ProPhoto	D50

The **GamutExtension** tag indicates whether or not to perform "gamut extension" when applying the color table transform. This tag must have a value of 0 or 1 with the following interpretation:

GamutExtension	*Meaning*
0	Do not perform gamut extension. Output pixel values will be clipped to the color space implied by the ColorPrimaries tag.
1	Perform gamut extension. For out-of-gamut pixels, compute the difference between the gamut-clipped values and the unclipped values; add the difference to the table-transformed result. Details on the gamut extension math are given below.

In general, it is recommended to use gamut extension (set the tag to 1).

Background Table

This tag supports the concept of a "background table" – namely, a color lookup table that can be applied to "background" pixels that are not covered by one of the other tables. Two examples:

Example 1: portrait and background

A DNG file represents a photograph of a person. The goal is to define two separate color transforms: one for the person and another for the background. Steps to implement this:

- Store a semantic mask in the file with the SemanticName tag set to "portrait_subject"
- Store a RGBTables tag in the file with two tables:
    - The first table has TableSemanticName set to "portrait_subject" and defines the color transform to be applied to the person
    - The second table has an empty TableSemanticName (LengthTableSemanticName set to 0) and defines the color transform to be applied to the background

Notes on the above example:

- The actual value of the SemanticName tag is not important. The key is to use the same value for SemanticName and TableSemanticName so that the color transform can be matched to that mask.
- When using the Weighted Sum composite method, the order of the two tables within the RGBTables tag is not important. When using the Sequential composite method, the portrait table will be applied first, then the background table, because this is the order they are specified in the tag.

Example 2: two portraits, sky, and background

A DNG file represents a photograph of two people in an outdoor setting with the sky visible. The goal is to define four separate color transforms: one for each person, one for the sky, and one for the background. Steps to implement this:

- Store semantic masks in the file for each person and the sky, with appropriate SemanticName tag values such as "subject1" "subject2" and "sky"
- Store a RGBTables tag in the file with four tables:
    - The first table has TableSemanticName set to "subject1" and defines the color transform to be applied to the first person
    - The second table has TableSemanticName set to "subject2" and defines the color transform to be applied to the second person
    - The third table has TableSemanticName set to "sky" and defines the color transform to be applied to the sky
    - The fourth table has an empty TableSemanticName (LengthTableSemanticName set to 0) and defines the color transform to be applied to the background

Note that at most one table in this tag can be used as the background table. That is, at most one table can use a zero value for LengthTableSemanticName. If no tables specify a zero value for LengthTableSemanticName, then there is no background table.

Table transform pipeline

This section describes the table transform pipeline using the GammaEncoding, ColorPrimaries, and GamutExtension tags.

Let **original_space** be the input RGB color space (before applying the table). In most cases this will be a standard RGB space such as sRGB or ProPhoto RGB.

Let **table_space** be the linear RGB color space specified by the ColorPrimaries tag (ignoring gamma encoding).

Let **table_gamma** be the gamma encoding curve specified by the GammaEncoding tag.

The steps are:

1. If needed, decode gamma for original_space. For example, if original_space is sRGB, then this step applies the sRGB decoding curve, so that the result pixel values are in "linear gamma" sRGB.
2. Apply 3x3 matrix transform to table_space.
3. If the camera profile is High Dynamic Range (see ProfileDynamicRange), then apply the encoding function $f(x)$ to each component.
4. Clamp each component to [0, 1]
5. Compute Delta =
    - difference between Step 4 and Step 3 (if GamutExtension is 1)
    - 0 (if GamutExtension is 0)
6. Apply GammaEncoding curve.
7. Apply 3D table.
8. Apply inverse of Step 6 (undo GammaEncoding curve).
9. Add Delta computed in Step 5.
10. If the camera profile is High Dynamic Range, then apply the decoding function $f^{-1}(x)$ to each component.
11. Apply inverse of Step 2 (transform back to original_space with linear gamma).

Notes on applying the color transforms

There are two methods for applying color transforms: Weighted Sum Method and Sequential Method. The method is specified using the CompositeMethod field.

Weighted Sum Method

The overall color transform is a linear combination of the tables, weighted by their corresponding Semantic Masks and a special weight for the Background Table (if any).

To make this precise, consider the following definitions:

- Let N be the number of tables, comprising one Background Table and $N-1$ tables with Semantic Masks
- Let T_i be the ith color table transform with a Semantic Mask, where i ranges from 1 to $N-1$
- Let B be the Background color table transform, or the identity transform if there is no Background Table
- Let m_i be the Semantic Mask corresponding to T_i
- Let $pSrc$ be the source color of a given pixel

To compute the final pixel color $pResult$, use the following equation:

$$pResult = \frac{\sum_{i=0}^{N-1}\big(m_i * T_i(pSrc)\big) + \big(b \cdot B(pSrc)\big)}{\sum_{i=0}^{N-1} m_i + b}$$

where

$$b = 1 - \min\left(1, \sum_{i=0}^{N-1} m_i\right)$$

Note that if all the Semantic Masks have zero weight (all m_i are zero) for a given pixel, then the background weight b will be 1. Furthermore, if there is no background table (B is the identity transform), then $pResult = pSrc$.

Sequential Method

The overall color transform is the result of applying each table sequentially (i.e., in an ordered list). The order of application is the same as the order of the tables specified in the tag. That is, the first table specified is the first table applied, the second table specified is the second table applied, and so on.

More precisely, using the same definitions as above:

$$p_{i+1} = p_i + m_i(T(p_i) - p_i)$$

For example, if there are 3 tables ($N = 3$), then:

$$p_0 = pSrc$$

$$p_1 = p_0 + m_0(T(p_0) - p_0)$$

$$p_2 = p_1 + m_1(T(p_1) - p_1)$$

$$p_3 = p_2 + m_2(T(p_2) - p_2)$$

$$pResult = p_3$$

The background table is optional; it may appear at most once in the list. The mask value b of the background table is the same as defined in the Weighted Sum Method. For example, continuing the above 3-table example, if the first table is the background table, then

$$p_1 = p_0 + b(B(p_0) - p_0)$$

Other Notes

Table entries should be interpolated using a smooth method, such as trilinear or tetrahedral interpolation.

The color transforms should be applied after the ProfileToneCurve tag.

This tag was originally introduced in DNG 1.6 with usage in IFD 0 only.

Starting with DNG 1.7, usage is allowed in both IFD 0 and Camera Profile IFD. When a reader is rendering a raw file with a camera profile, the profile's RGBTables tag takes precedence of any RGBTables in IFD 0.

CalibrationIlluminant3

Tag 52529 (CD31.H)

Type SHORT

Count 1

Value See below

Default 0 (unknown)

Usage IFD 0 or Camera Profile IFD

Description

The illuminant used for the third set of color calibration tags. The legal values for this tag are the same as the legal values for the LightSource EXIF tag. The value may be set to 255 (Other) to indicate a custom illuminant. If set to 255, then the IFD must also include a IlluminantData3 tag to specify the white point or spectral data for this illuminant.

See Chapter 6, "Mapping Camera Color Space to CIE XYZ Space" for details of the color-processing model, and extra requirements related to triple-illuminant camera profiles.

ColorMatrix3

Tag 52531 (CD33.H)

Type SRATIONAL

Count ColorPlanes * 3

Value See below

Default None

Usage IFD 0 or Camera Profile IFD

Description

ColorMatrix3 defines a transformation matrix that converts XYZ values to reference camera native color space values, under the third calibration illuminant. The matrix values are stored in row scan order.

See Chapter 6, "Mapping Camera Color Space to CIE XYZ Space" for details of the color-processing model, and extra requirements related to triple-illuminant camera profiles.

CameraCalibration3

Tag 52530 (CD32.H)

Type SRATIONAL

Count ColorPlanes * ColorPlanes

Value See below

Default Identity matrix

Usage IFD 0

Description

CameraCalibration3 defines a calibration matrix that transforms reference camera native space values to individual camera native space values under the third calibration illuminant. The matrix is stored in row scan order.

This matrix is stored separately from the matrix specified by the ColorMatrix3 tag to allow raw converters to swap in replacement color matrices based on UniqueCameraModel tag, while still taking advantage of any per-individual camera calibration performed by the camera manufacturer.

See Chapter 6, "Mapping Camera Color Space to CIE XYZ Space" for details of the color-processing model, and extra requirements related to triple-illuminant camera profiles.

ReductionMatrix3

Tag 52538 (CD3A.H)

Type SRATIONAL

Count ColorPlanes * 3

Value See below

Default None

Usage IFD 0 or Camera Profile IFD

Description

ReductionMatrix3 defines a dimensionality reduction matrix for use as the first stage in converting color camera native space values to XYZ values, under the third calibration illuminant. This tag may only be used if ColorPlanes is greater than 3. The matrix is stored in row scan order.

See Chapter 6, "Mapping Camera Color Space to CIE XYZ Space" for details of the color-processing model, and extra requirements related to triple-illuminant camera profiles.

ProfileHueSatMapData3

Tag 52537 (CD39.H)

Type FLOAT

Count HueDivisions * SaturationDivisions * ValueDivisions * 3

Value See below

Default None

Usage IFD 0 or Camera Profile IFD

Description

This tag contains the data for the third hue/saturation/value mapping table. Each entry of the table contains three 32-bit IEEE floating-point values. The first entry is hue shift in degrees; the second entry is saturation scale factor; and the third entry is a value scale factor. The table entries are stored in the tag in nested loop order, with the value divisions in the outer loop, the hue divisions in the middle loop, and the saturation divisions in the inner loop. All zero input saturation entries are required to have a value scale factor of 1.0. The hue/saturation/value table application and requirements are described in detail in Chapter 6.

ForwardMatrix3

Tag 52532 (CD34.H)

Type SRATIONAL

Count ColorPlanes * 3

Value See below

Default None

Usage IFD 0 or Camera Profile IFD

Description

This tag defines a matrix that maps white balanced camera colors to XYZ D50 colors. Application and requirements are described in detail in Chapter 6.

IlluminantData1

Tag 52533 (CD35.H)

Type UNDEFINED

Count See below

Value See below

Default None

Usage IFD 0 or Camera Profile IFD

Description

When the CalibrationIlluminant1 tag is set to 255 (Other), then the IlluminantData1 tag is required and specifies the data for the first illuminant. Otherwise, this tag is ignored.

The illuminant data may be specified as either a x-y chromaticity coordinate or as a spectral power distribution function.

The format of the tag data is as follows.

The first 2 bytes is a SHORT integer describing the data type. A value of 0 means the data is a x-y chromaticity coordinate. A value of 1 means the data is a spectral power distribution function.

If the data type is 0 (x-y chromaticity coordinate), then the tag format is:

DataType - SHORT (0)

x - RATIONAL

y - RATIONAL

If the data type is 1 (spectral power function), then the tag format is:

DataType - SHORT (1)

NumLambda - LONG

MinLambda - RATIONAL

LambdaSpacing - RATIONAL

For each NumLambda

Spectral power distribution function value - RATIONAL

MinLambda is the wavelength of the first sample, in nanometers, and should typically be between 360 and 400.

LambdaSpacing is the spacing between adjacent samples, in nanometers.

NumLambda is the total number of samples. It must be at least 2.

It is recommended that the provided spectral data cover at least 360 nm to 830 nm.

For example, if MinLambda is 360, LambdaSpacing is 10, and NumLambda is 48, then the spectral power distribution function is provided from 360 to 830 nm in 10 nm increments. The first sample is at 360 nm, the 2nd is at 370 nm, the 3rd is at 380 nm, and so on; the last sample is at 830 nm.

The x-y chromaticity coordinate of the illuminant is calculated by integrating the provided spectral power distribution with the CIE 2-degree standard observer functions from 360 nm to 830 nm. Wavelengths below the minimum lambda value (MinLambda) map to the first provided spectral value; wavelengths above the

maximum lambda value map to the last provided spectral value. Wavelengths in between adjacent sample values are mapped using linear interpolation.

IlluminantData2

Tag 52534 (CD36.H)

Type UNDEFINED

Count See below

Value See below

Default None

Usage IFD 0 or Camera Profile IFD

Description

When the CalibrationIlluminant2 tag is set to 255 (Other), then the IlluminantData2 tag is required and specifies the data for the second illuminant. Otherwise, this tag is ignored.

The format of the data is the same as IlluminantData1.

IlluminantData3

Tag 52535 (CD37.H)

Type UNDEFINED

Count See below

Value See below

Default None

Usage IFD 0 or Camera Profile IFD

Description

When the CalibrationIlluminant3 tag is set to 255 (Other), then the IlluminantData3 tag is required and specifies the data for the third illuminant. Otherwise, this tag is ignored.

The format of the data is the same as IlluminantData1.

Additional Tags for Version 1.7.0.0

The following tags have been added for the 1.7.0.0 version of this specification.

ProfileGainTableMap2

Tag 52544 (CD40.H)

Type UNDEFINED

Count Byte count of data; see below

Value See below

Default Optional

Usage IFD 0 or Camera Profile IFD

Description

This tag is an extended version of ProfileGainTableMap. It consists of the following parameters, in order:

MapPointsV - LONG

MapPointsH - LONG

MapSpacingV - DOUBLE

MapSpacingH - DOUBLE

MapOriginV - DOUBLE

MapOriginH - DOUBLE

MapPointsN - LONG

MapInputWeights - FLOAT × 5 (e.g., 1/3, 1/3, 1/3, 0, 0)

DataType - LONG

Gamma - FLOAT

GainMin - FLOAT

GainMax - FLOAT

For Each MapPointsV

For Each MapPointsH

For Each MapPointsN

Gain value - UINT8, UINT16, FLOAT16, or FLOAT (see below)

The tag count is the sum of the byte count of all the parameters listed above. This value is 80 + (B * MapPointsV * MapPointsH * MapPointsN), where

- B is 1 if DataType is 0 (unsigned 8-bit integer)
- B is 2 if DataType is 1 or 2 (unsigned 16-bit integer or 16-bit floating-point)
- B if 4 if DataType is 3 (32-bit floating point)

For example, if MapPointsV = 6, MapPointsH = 8, and MapPointsN = 256, and DataType is 0 (8-bit unsigned integer), then the total tag size is 80 + (1 * 6 * 8 * 256) = 12368 bytes.

This tag is the same as the existing ProfileGainTableMap, with the following changes:

- Addition of the DataType, Gamma, GainMin, and GainMax parameters (see below for details)
- The gain values can be specified using integers or floating-point values

DataType

DataType must be one of the following values:

DataType	Storage type of gain value	Notes
0	8-bit unsigned integer	Must use GainMin and GainMax
1	16-bit unsigned integer	Must use GainMin and GainMax
2	16-bit floating point	GainMin and GainMax are ignored
3	32-bit floating point	GainMin and GainMax are ignored

Gamma

Gamma is a parameter that is applied to the weighted input value prior to table lookup (see details below). A value of 1.0 is a NOP (no change). The minimum and maximum allowed gamma values are 0.25 and 4.0, respectively.

GainMin and GainMax

For the case where DataType is 0 or 1 (integer representation of gain values), GainMin and GainMax represent the minimum and maximum 32-bit floating-point values, respectively. For example, if GainMin is 0.7 and GainMax is 2.3, and the DataType is 0 (8-bit unsigned integer), then a stored 8-bit value of 0 represents the minimum gain of 0.7, and the maximum stored 8-bit value of 255 represents the maximum gain value of 2.3.

Application

At each point in the image, up to four gain tables are interpolated to determine the gain table to apply at that location. Given the resulting interpolated table, the following method is used to produce an output RGB color given the input RGB color at that point in the image:

1. The “table input value” is a scalar value calculated using the following equation:

   table input value = pow (clamp ((R, G, B, min (R,G,B), max(R,G,B)) · MapInputWeights, 0, 1), Gamma)

2. Next, the “gain value” **G** is calculated by using “table input value” to perform a linearly interpolated lookup of the gain table.

3. If DataType is 0 (8-bit unsigned integer) or 1 (16-bit unsigned integer), then the value obtained in Step 2 is mapped to the floating-point gain using the GainMin and GainMax parameters as follows:

   FloatGain = GainMin + (**G** / MaxInt) * (GainMax - GainMin)

   where MaxInt is 255.0 if DataType is 0 and 65535.0 if DataType is 1.

4. The above gain is multiplied by the input RGB camera values to produce new RGB values. The resulting RGB values may be overrange (greater than 1). DNG readers are not required to clip the values to the nominal [0,1] range; readers are encouraged to preserve the overrange values for potential use in downstream tone adjustments.

Compatibility with ProfileGainTableMap

Both ProfileGainTableMap and ProfileGainTableMap2 tags may be present. If both are present, then ProfileGainTableMap2 supersedes ProfileGainTableMap; that is, DNG readers should apply only ProfileGainTableMap2 and ignore ProfileGainTableMap. This behavior is intended for compatibility reasons.

If ProfileGainTableMap2 is used in a Camera Profile IFD and that profile is used to render a raw file, the profile-specific ProfileGainTableMap2 should be used instead of a ProfileGainTableMap2 specified in the Raw IFD.

IFD 0 vs Camera Profile IFD

ProfileGainTableMap2 can be used in either IFD 0 or Camera Profile IFD. The latter option allows ProfileGainTableMap2 to be a per-camera profile property.

When a raw file is rendered using that profile, that profile's ProfileGainTableMap2 should be used instead of any ProfileGainTableMap2 specified in IFD 0.

When a raw file is rendered using a profile that does not specify a ProfileGainTableMap2, then any ProfileGainTableMap2 in IFD 0 or ProfileGainTableMap in Raw IFD should be used.

Summary of the precedence order for choosing which data to use:

- ProfileGainTableMap2 in Camera Profile IFD (highest precedence)
- ProfileGainTableMap2 in IFD 0
- ProfileGainTableMap in Raw IFD (lowest precedence)

IFD 0 vs Raw IFD

The intent of the original ProfileGainTableMap tag (introduced in DNG 1.6) was to use IFD 0. However, the DNG 1.6 specification mistakenly indicated "Raw IFD" instead of IFD 0. This is an error, because the same gain table map operation is meaningful for all scene-referred IFDs in the file, including raw previews and Enhanced IFDs (not only the raw IFD).

ProfileGainTableMap2 (introduced in DNG 1.7) uses IFD 0 or Camera Profile IFD.

ImageSequenceInfo

Tag 52548 (CD44.H)

Type UNDEFINED

Count Byte count of data

Value See below

Default Optional

Usage IFD 0

Description

This is an informative tag that describes how the image file relates to other image files captured in a sequence. Applications include focus stacking, merging multiple frames to reduce noise, time lapses, exposure brackets, stitched images for super resolution, and so on.

The following table describes the fields in this tag, in order:

Field	Type	Description
Sequence ID	Null-terminated ASCII string at least 8 characters	Unique identifier of the image sequence. All image files in the sequence should use the same string. It is recommended that DNG writers use a prefix unique to the author or company, e.g., prefix “com.adobe” as part of a broader identifier “com.adobe.F92C483A22B7”
Sequence Type	Null-terminated ASCII string at least 1 character	Type of image sequence. All image files in this sequence should use the same string. See below for details.
Frame Info	Null-terminated ASCII string	Information specific to this image within the sequence. This string may be empty, and its contents may vary from image to image within the sequence.
Index	32-bit unsigned integer, big-endian	The 1-based index of the image within the sequence (i.e., the first frame in the sequence has index value 1). A value of 0 means the index is unknown or not available.
Count	32-bit unsigned integer, big-endian	The total number of images in the sequence. A value of 0 means the count is unknown or not available.
Final	8-bit unsigned integer	Indicator of final image in the sequence: • 0 = no, this is not the final image • 1 = yes, this is the final image • other = unknown if this is the final image

The Sequence Type field should be a human-readable string describing the purpose of the image sequence. It may be vendor-specific or camera model-specific.

Example values include:

- Exposure Bracket
- Focus Stack
- Timelapse
- CFA Shift
- Panorama Segment
- Burst Denoise

ImageStats

Tag 52550 (CD46.H)

Type UNDEFINED

Count Byte count of data

Value See below

Default Optional

Usage Raw IFD

Description

This is an informative tag that provides basic statistical information about the pixel values of the image in this IFD. Possible applications include normalizing brightness of images when multiple images are displayed together (especially when mixing Standard Dynamic Range and High Dynamic Range images), identifying underexposed or overexposed images, and so on.

All floating-point values in this tag use a normalized convention where 0.0 corresponds to black and 1.0 corresponds to white. The statistics describe the state of the image after mapping to linear reference values (as described in this chapter), so that the black level is mapped to 0.0 and the white level is mapped to 1.0.

The data in this tag is stored in big-endian byte order.

The format of this tag data is:

LONG – Number of child tags (N)

Child tag table

The child tag table is a tightly-packed sequence of N child tags, where N is the number of child tags. The format of each child tag is:

LONG - Child tag code

LONG – Byte length of child tag data (L)

Child tag data (L bytes)

The set of child tags is described in the following table:

Child Tag Code	Byte Length	Child Tag Data	Description
1	4	32-bit float	Weighted average pixel value
2	4 + (8 * M)	See below	List of M weighted samples
3	4 * P	P 32-bit floats	Weight values used to compute weighted average and samples (child tags 1 and 2)
4	4 * P	P 32-bit floats	Per-color plane average pixel values
5	4 + (M * 4 * (P + 1))	See below	List of M per-color plane samples

For child tags 3 through 5, the variable P represents the number of color planes. It must match the number of color planes for the image in the IFD that includes this tag.

Child tag 1 is the weighted average pixel value. For a color image, the weighted average is computed using a linear combination of pixel values. For example, for RGB data where P is 3:

$$y = k_1 r + k_2 g + k_3 b$$

where k_1, k_2, and k_3 are the weights described in child tag 3. The same weighting formula is used for child tag 2 (list of weighted samples).

Child tag 4 is similar to child tag 1, except that the average pixel values are specified separately for each color plane. For monochrome images, child tags 1 and 4 are logically equivalent.

Child tags 2 and 5 specify a list of samples associated with an order. They can be used to provide ordered statistics similar to a histogram. The difference between these tags is that child tag 2 uses weighted pixel values (using the weights described in child tag 3), whereas child tag 5 uses per-color plane values.

The data format for child tag 2 is:

LONG – number of samples M

FLOAT – fractional position F_1 for sample 1

FLOAT – weighted value for sample 1

FLOAT – fractional position F_2 for sample 2

FLOAT – weighted value for sample 2

…

FLOAT – fractional position F_M for sample M

FLOAT – weighted value for sample M

The data format for child tag 5 is similar but uses per-color plane (not weighted) values:

LONG – number of samples M

FLOAT – fractional position F_1 for sample 1

FLOAT – value for color plane 1 of sample 1

…

FLOAT – value for color plane P of sample 1

…

FLOAT – fractional position F_M for sample M

FLOAT – value for color plane 1 of sample M

…

FLOAT – value for color plane P of sample M

The "fractional position" F of a given sample is a value in [0,1] that describes the relative position of that sample, compared to the values of other samples in the image.

Let S be the total number of pixels in the image. Sort the pixels of a single color plane (or weighted color plane) in the image in ascending (increasing order). The first pixel in the sorted list has the minimum value. The last pixel in the sorted list has the maximum value. Given a pixel at position N in the sorted list, where N is in [0, S-1], define the fractional position F as:

$$F = \frac{N}{S-1}$$

Examples:

- F = 0.0 corresponds to the first sorted pixel (N = 0). It is the minimum pixel value.
- F = 1.0 corresponds to the last sorted pixel (N = S - 1). It is the maximum pixel value.
- F = 0.5 corresponds (approximately) to the middle pixel. It is the median pixel value.

Notes and restrictions on child tags 2 and 5 (list of samples):

- There must be at least 1 sample ($M \geq 1$) per list.
- The maximum number of samples per list is 1024 ($M \leq 1024$).
- The samples must appear in order of strictly ascending (increasing) values of F. No fractional position value may be repeated. For instance, if a list has 3 samples with fractional positions F_1, F_2, and F_3, then it must be that $F_1 < F_2 < F_3$.
- For monochrome images, child tags 2 and 5 are logically identical.

For child tags 4 and 5, which describe per-color plane information, the order of the data values follows the plane order specified by the CFAPlaneColor tag (for CFA images).

All child tags are optional and may be included in any combination. For example, a writer may choose to include only child tag 1 (weighted average pixel value), without indicating what the weight values are (i.e., omitting child tag 3).

Child tags may appear in any order in the child tag table.

Child tags may not be repeated. That is, a child tag may appear at most once in a table.

An empty child tag table (N = 0) is permitted.

For performance reasons, DNG writers may compute the values of this tag using downsampled (lower-resolution) images. DNG readers should assume the values in this tag are approximate.

ProfileDynamicRange

Tag 52551 (CD47.H)

Type UNDEFINED

Count 8

Value See below

Default Optional (Standard Dynamic Range)

Usage IFD 0 or Camera Profile IFD

Description

This tag describes the intended rendering output dynamic range for a given camera profile.

If this tag is omitted, then it is assumed that the intended output dynamic range of the profile is Standard Dynamic Range (SDR). That is, the rendered output pixels are limited to the range [0, 1], where 1.0 is the "user interface white level" (UI White) of the display.

The data format of the tag is:

SHORT – Version

SHORT – DynamicRange

FLOAT – HintMaxOutputValue

The Version field describes the version of this tag. It must have the value 1.

DynamicRange indicates the intended output dynamic range of the profile. Supported values are:

0 – Standard Dynamic Range

1 – High Dynamic Range

A value of 1 (High Dynamic Range) indicates that the given camera profile should be applied in High Dynamic Range mode. There are two aspects of this behavior.

First, the final rendered pixel values may have values outside of [0, 1]. For example, an output pixel value of 4.0 appears 4 times as bright as UI White, assuming a linear gamma encoding of the image. Implementations should avoid prematurely clipping intermediate pixel values within their rendering pipelines to [0, 1].

Second, the math for applying several of the camera profile tags is adjusted for HDR camera profiles to accommodate overrange values. These tags are:

- ProfileHueSatMapData[1,2,3]
- ProfileHueSatMapEncoding
- ProfileLookTableData
- ProfileLookTableEncoding
- ProfileToneCurve

- RGBTables

For each of the above tags, the general method for applying them to HDR data is:

- Apply an encoding function $f(x)$ to each color component.
- Apply the tag normally (same as in SDR mode), but without clipping the results to [0, 1].
- Apply a decoding function $f^{-1}(x)$ to each color component.

The encoding function $f(x)$ is:

$$f(x) = \frac{x(256 + x)}{256(1 + x)}$$

The decoding function $f^{-1}(x)$ is the inverse of $f(x)$:

$$f^{-1}(x) = 16\left(8x - 8 + \sqrt{64x^2 - 127x + 64}\right)$$

The encoding function smoothly compresses the overrange domain [0, 16] to a normalized range [0, 1]. This provides a mechanism for lookup tables and curves (which are normally defined on a limited range [0, 1]) to operate on overrange data. More details on applying the tags for High Dynamic Range profiles are given in the descriptions of each tag and in the section on Applying the Hue/Saturation/Value Mapping Table.

HintMaxOutputValue is a hint that describes the intended maximum output pixel value for the given camera profile, expressed using a linear gamma encoding (gamma 1.0). For example, a value of 8.0 indicates that the intended rendered output pixel values should not exceed 8.0 (or 3 f-stops above UI White). When DynamicRange is 0 (SDR), HintMaxOutputValue must be ≤ 1. A value less than or equal to 0 indicates no hint (unknown).

ProfileGroupName

Tag 52552 (CD48.H)

Type ASCII or BYTE

Count Length of string including null

Value Null-terminated string

Default Optional

Usage IFD 0 or Camera Profile IFD

Description

A UTF-8 encoded string containing the “group name” of the camera profile. The purpose of this tag is to associate two or more related camera profiles into a common group. A potential application of this tag is to relate two profiles, one of which is intended for Standard Dynamic Range (SDR) rendering and the other of which is intended for High Dynamic Range (HDR) rendering. The reader can select which profile within the group is appropriate based on the desired output dynamic range.

For example, a DNG file could contain 4 embedded profiles as follows:

ProfileName	**ProfileDynamicRange**	**ProfileGroup**	**Notes**
Portrait SDR	SDR (DynamicRange = 0)	Portrait	A camera profile optimized for portrait photos, designed for SDR display.
Portrait HDR	HDR (DynamicRange = 1)	Portrait	A camera profile optimized for portrait photos, designed for HDR display.
Landscape SDR	SDR (DynamicRange = 0)	Landscape	A camera profile optimized for landscape photos, designed for SDR display.
Landscape HDR	HDR (DynamicRange = 1)	Landscape	A camera profile optimized for landscape photos, designed for HDR display.

Additional Tags for Version 1.7.1.0

The following tags have been added for the 1.7.0.0 version of this specification.

ColumnInterleaveFactor

Tag 52547 (CD43.H)

Type SHORT or LONG

Count 1

Value ColumnInterleaveFactor

Default 1

Usage Raw IFD

Description

This tag specifies that columns of the image are stored in interleaved order. The value of the tag specifies the number of interleaved fields. The use of a non-default value for this tag requires setting the DNGBackwardVersion tag to at least 1.7.1.0.

Potential Usage

This tag can be used in combination with RowInterleaveFactor to store Color Filter Array (mosaic) image data as a set of subimages. This may be useful when using compression methods that do not natively support interleaved pixels.

For example, a Bayer mosaic image with a 2x2 CFA pattern can be stored as a set of four monochrome subimages, each of which are ½ the width and ½ the height of the full image, by setting both RowInterleaveFactor and ColumnInterleaveFactor to 2. The top-left color of the 2x2 pattern maps to the top-left subimage (e.g., red), the top-right color maps to the top-right subimage (e.g., green), and so on. Each subimage can be compressed independently.

JXLDistance

Tag 52553 (CD49.H)

Type FLOAT

Count 1

Value JPEG XL distance parameter

Default Optional

Usage IFD using JXL compression

Description

This optional tag specifies the distance parameter used to encode the JPEG XL data in this IFD. A value of 0.0 means lossless compression, while values greater than 0.0 means lossy compression.

JXLEffort

Tag 52554 (CD4A.H)

Type LONG

Count 1

Value JPEG XL effort parameter

Default Optional

Usage IFD using JPEG XL compression

Description

This optional tag specifies the effort parameter used to encode the JPEG XL data in this IFD. Values range from 1 (low) to 9 (high).

JXLDecodeSpeed

Tag 52555 (CD4B.H)

Type LONG

Count 1

Value JPEG XL decode speed parameter

Default Optional

Usage IFD using JXL compression

Description

This optional tag specifies the decode speed parameter used to encode the JPEG XL data in this IFD. Values range from 1 (slow) to 4 (fast).

Mapping Raw Values to Linear Reference Values

The section describes DNG's processing model for mapping stored raw sensor values into linear reference values.

Linear reference values encode zero light as 0.0, and the maximum useful value (limited by either sensor saturation or analog to digital converter clipping) as 1.0. If SamplesPerPixel is greater than one, each sample plane should be processed independently.

The processing model follows these steps:

- Linearization
- Black Subtraction
- Rescaling
- Clipping

Linearization

The first step is to process the raw values through the look-up table specified by the LinearizationTable tag, if any. If the raw value is greater than the size of the table, it is mapped to the last entry of the table.

Black Subtraction

The black level for each pixel is then computed and subtracted. The black level for each pixel is the sum of the black levels specified by the BlackLevel, BlackLevelDeltaH and BlackLevelDeltaV tags.

Rescaling (Normalization)

The black subtracted values are then rescaled to map them to a logical 0.0 to 1.0 range. The scale factor is the inverse of the difference between the value specified in the WhiteLevel tag and the maximum computed black level for the sample plane.

Clipping

Rescaled values above 1.0 should be clipped to 1.0. Rescaled values below 0.0 may be clipped to 0.0, but it is recommended to preserve negative values for at least the early stages of an image rendering pipeline (e.g., for better noise reduction in the shadows).

Mapping Camera Color Space to CIE XYZ Space

This section describes the DNG processing model for mapping between the camera color space coordinates (linear reference values) and CIE XYZ (with a D50 white point).

Camera Calibration Matrices

DNG 1.2.0.0 and later supports different companies creating the camera calibration tags using different reference cameras.

When rendering a DNG file using a camera profile, it is important to know if the selected camera profile was designed using the same reference camera used to create the camera calibration tags. If so, then the camera calibration tags should be used. If not, then it is preferable to ignore the camera calibration tags and use identity matrices instead in order to minimize the worst case calibration mismatch error.

This matching is done by comparing the CameraCalibrationSignature tag and the ProfileCalibrationSignature tag for the selected camera profile. If they match, then use the camera calibration tags. If not, then use identity matrices.

One, Two, or Three Color Calibrations

DNG provides for one, two, or three sets of color calibration tags, each set optimized for a different illuminant. If two or three sets of color calibration tags are included, then the raw converter should interpolate between the calibrations based on the white balance selected by the user.

Information for two calibrations

If exactly two calibrations are included, then it is recommended that one of the calibrations be for a low color temperature illuminant (e.g., Standard-A) and the second calibration illuminant be for a higher color temperature illuminant (e.g., D55 or D65). This combination has been found to work well for a wide range of real-world digital camera images.

DNG versions earlier than 1.2.0.0 allow the raw converter to choose the interpolation algorithm. DNG 1.2.0.0 and later requires a specific interpolation algorithm: linear interpolation using inverse correlated color temperature. To find the interpolation weighting factor between the two tag sets, find the correlated color temperature for the user-selected white balance and the two calibration illuminants. If the white balance temperature is between two calibration illuminant temperatures, then invert all the temperatures and use linear interpolation. Otherwise, use the closest calibration tag set.

Information for three calibrations

If three calibrations are included, then it is recommended that the first two calibrations are as described in the preceding paragraph, and the third calibration be used to characterize the "as shot" illuminant: that is, the illuminant used to photograph the main image. This third calibration provides a color transform optimized for the capture conditions, while providing additional calibrations for two other illuminants in case the user wishes to change the white balance significantly during raw conversion. Note that the 3rd calibration is introduced in DNG version 1.6.0.0.

Requirements for three calibrations

If CalibrationIlluminant3 is included, then the following are required:

- CalibrationIlluminant1 and CalibrationIlluminant2 must be included.
- ColorMatrix3 must be included.
- Either all three ForwardMatrix tags are included, or none of them are included.
- Either all three ReductionMatrix tags are included, or none of them are included.
- Either all three ProfileHueSatMapData tags are included, or none of them are included.
- The white points (x-y chromaticity values) of all three calibration illuminants must be distinct.

Definitions used in the following sections

Let n be the dimensionality of the camera color space (usually 3 or 4).

Let CM be the n-by-3 matrix interpolated from the ColorMatrix1, ColorMatrix2, and ColorMatrix3 tags.

Let CC be the n-by-n matrix interpolated from the CameraCalibration1, CameraCalibration2, and CameraCalibration3 tags (or identity matrices, if the signatures don't match).

Let AB be the n-by-n matrix, which is zero except for the diagonal entries, which are defined by the AnalogBalance tag.

Let RM be the 3-by-n matrix interpolated from the ReductionMatrix1, ReductionMatrix2, and ReductionMatrix3 tags.

Let FM be the 3-by-n matrix interpolated from the ForwardMatrix1, ForwardMatrix2, and ForwardMatrix3 tags.

Translating White Balance xy Coordinates to Camera Neutral Coordinates

If the white balance is specified in terms of a CIE xy coordinate, then a camera neutral coordinate can be derived by first calculating the weighting factor between the two or three sets of color calibration tags.

The XYZ to camera space matrix is:

XYZtoCamera = AB * CC * CM

The camera neutral can be found by expanding the xy value to a 3-by-1 XYZ matrix (assuming Y = 1.0) and multiplying it by the XYZtoCamera matrix:

CameraNeutral = XYZtoCamera * XYZ

Translating Camera Neutral Coordinates to White Balance xy Coordinates

This process is slightly more complex than the transform in the other direction because it requires an iterative solution.

1. Guess an xy value. Use that guess to find the interpolation weighting factor between the color calibration tags. Find the XYZtoCamera matrix as above.
2. Find a new xy value by computing:

   XYZ = Inverse (XYZtoCamera) * CameraNeutral

   (If the XYZtoCamera matrix is not square, then use the pseudo inverse.)
3. Convert the resulting XYZ to a new xy value.
4. Iterate until the xy values converge to a solution.

Camera to XYZ (D50) Transform

DNG 1.2.0.0 and later support two methods of specifying the camera to XYZ (D50) transform, depending on whether or not the forward matrix tags are included in the camera profile.

The use of the forward matrix tags is recommended for two reasons. First, it allows the camera profile creator to control the chromatic adaptation algorithm used to convert between the calibration illuminant and D50. Second, it causes the white balance adjustment (if the user white balance does not match the calibration illuminant) to be done by scaling the camera coordinates rather than by adapting the resulting XYZ values, which has been found to work better in extreme cases.

If the ForwardMatrix tags are not included in the camera profile

1. First, invert the XYZ to Cameramatrix.

   If n = 3, this is:

   CameraToXYZ = Inverse (XYZtoCamera)

   If n > 3, and the reduction matrix tags are included, then:

   CameraToXYZ = Inverse (RM * XYZtoCamera) * RM

   Otherwise:

   CameraToXYZ = PseudoInverse (XYZtoCamera)

2. The white balanced transform is computed:

   CameraToXYZ_D50 = CA * CameraToXYZ

   CA, above, is a chromatic adaptation matrix that maps from the white balance xy value to the D50 white point. The recommended method for computing this chromatic adaptation matrix is to use the linear Bradford algorithm.

If the ForwardMatrix tags are included in the camera profile

CameraToXYZ_D50 = FM * D * Inverse (AB * CC)

D, above, is a diagonal n-by-n matrix, computed so that the CameraToXYZ_D50 matrix maps the selected camera neutral to XYZ D50. The forward matrix is required to map a unit vector to XYZ D50 by definition, so D can be computed by finding the neutral for the reference camera:

ReferenceNeutral = Inverse (AB * CC) * CameraNeutral

And then:

D = Invert (AsDiagonalMatrix (ReferenceNeutral))

Applying the Hue/Saturation/Value Mapping Table

After the camera colors have been converted to XYZ (D50) values, the Hue/Saturation/Value mapping table, if any, is applied. If there are two or three Hue/Saturation/Value mapping tables, then they are interpolated in the same way that color calibration tags are interpolated. If only one Hue/Saturation/Value table is included, then it is used regardless of the selected white balance.

1. First, the XYZ (D50) values are converted to linear RGB coordinates, using the ProPhoto RGB primaries. (This is also known as RIMM space).
2. If the camera profile is High Dynamic Range (see ProfileDynamicRange) and ValueDivisions is greater than 1, then apply the encoding function $f(x)$ to each color component.
3. The linear RGB coordinates are converted to HSV coordinates (Hue-Saturation-Value).
4. The HSV coordinates are used to index the mapping table using tri-linear interpolation, resulting in three values: hue shift (in degrees); saturation scale factor; value scale factor. If the division count in a dimension is 1, then the table is constant for that dimension. If the camera profile is High Dynamic Range, the input value (V) coordinate is clamped to [0, 1] prior to indexing the mapping table.
5. Hue is indexed using “wrap-around” math. For example, if HueDivisions is equal to 3, then the table samples are at 0 degrees (red), 120 degrees (green), and 240 degrees (blue).
6. The hue coordinate is modified by adding the hue shift.
7. The saturation coordinate is modified by multiplying by the saturation scale factor, and then clipping to no more than 1.0.
8. The value coordinate is modified by multiplying by the value scale factor. If the camera profile is Standard Dynamic Range, clip the result to no more than 1.0.
9. The HSV coordinates are converted to linear RGB coordinates.
10. If the camera profile is High Dynamic Range (see ProfileDynamicRange) and ValueDivisions is greater than 1, then applying decoding function $f^{-1}(x)$ (inverse of the encoding function) to each color component.
11. Convert from linear ProPhoto RGB back to XYZ (D50) values.

It is recommended that these tables be limited to use a ValueDivisions equal to 1, so the table is only indexed by hue and saturation. In this way, all colors with the same hue and saturation, but with different values, map to the same new hue and saturation while preserving their value ratios.

Special compatibility note with DNG 1.2

For maximum compatibility across DNG readers, it is recommended to use the first two calibration illuminants for standard illuminant types (such as A and D65) and to reserve the third illuminant for custom and "as shot" illuminants. DNG readers older than version 1.6 will be able to read and interpret the color

transforms (matrices and tables) from the first two illuminants. Newer DNG readers (DNG 1.6 and later) will be able to read and interpret the color transforms matrices and tables from all three illuminants.

Opcode List Processing

An opcode list is a list of opcodes, each of which does some specified image processing operation. Each opcode is performed in sequence.

Opcode lists are always stored in big-endian byte order, no matter what the file's main byte order is. This allows DNG utility programs to copy opcode lists from file to file, without needing to understand their detailed internal structure.

At the start of opcode list, there is a 32-bit unsigned integer count, which contains the number of opcodes in the list. This is followed by the data for each opcode.

Each opcode starts with a 32-bit unsigned integer, which contains the opcode ID. The opcode ID identifies the specific opcode. Documentation for each supported opcode ID in provided later in this chapter.

Next is a 32-bit unsigned integer, which contains the DNG specification version in which the opcode ID was defined. It is expected that new opcode IDs will be defined in future DNG specification versions. A DNG reader should never attempt to process an opcode with a version higher than DNG specification it was written to support.

Next is a 32-bit unsigned integer, which contains various flag bits. There are two defined flag bits. If bit 0 (the least significant bit) is set to 1, the opcode is considered optional, and the DNG reader may decide to not apply this opcode if it wishes, or it does not understand the opcode ID. If bit 1 (the second to least significant bit) is set to 1, the opcode can be skipped when doing "preview quality" processing, and only needs to be applied when doing "full quality" processing.

Next is a 32-bit unsigned integer, containing the number of bytes in a variable size parameter area for the opcode. The format of this variable size parameter area is dependent on the specific opcode ID, and is documented later in this chapter, along with each supported opcode ID.

When processing an opcode list, image values are clipped after the application of each opcode to the logical range of the image being modified. For OpcodeList1, this range is 0 to 2^{32}-1 for images with a bit depth greater than 16, otherwise 0 to 2^{16}-1. For OpcodeList2 and OpcodeList3, the logical range is 0.0 to 1.0.

As a special case, DNG readers should skip processing an opcode if both of the following conditions are met:

- it is a WarpRectilinear or WarpFisheye opcode, and
- the opcode immediately follows (in the same opcode list) a WarpRectilinear2 opcode that is marked as Optional

This "skip rule" is intended to maximize compatibility across DNG readers. See WarpRectilinear2 for more details.

The rest of this chapter contains information on each supported opcode ID.

WarpRectilinear

OpcodeID 1

DNG Version 1.3.0.0

Parameters

N LONG

For each coefficient set $i \in 1, 2, \ldots N$

$k_{r_0,i}$ DOUBLE

$k_{r_1,i}$ DOUBLE

$k_{r_2,i}$ DOUBLE

$k_{r_3,i}$ DOUBLE

$k_{t_0,i}$ DOUBLE

$k_{t_1,i}$ DOUBLE

$\hat{c}_x$ DOUBLE

$\hat{c}_y$ DOUBLE

Description

This opcode applies a warp to an image and can be used to correct geometric distortion and lateral (transverse) chromatic aberration for rectilinear lenses. The warp function supports both radial and tangential distortion correction.

Let $K_i = \{k_{r_0,i}, k_{r_1,i}, k_{r_2,i}, k_{r_3,i}, k_{t_0,i}, k_{t_1,i}\}$ denote the ith coefficient set, where $i \in 1, 2, \ldots, N$. Parameter N is the number of coefficient sets. N must be 1 or the total number of image planes. If N = 1, then a single set of warp coefficients (i.e., K_1) is applied to all image planes, i.e., all planes undergo the same transformation. If N > 1, coefficient set K_i is used to process the ith image plane, e.g., K_1 defines the warp function for the first image plane, K_2 defines the warp function for the second image plane, etc.

Parameters $K_i = \{k_{r_0,i}, k_{r_1,i}, k_{r_2,i}, k_{r_3,i}, k_{t_0,i}, k_{t_1,i}\}$ are the radial and tangential coefficients that define the warp function for the ith image plane; see below for implementation details and restrictions. Note that if $k_{r_0,i} = 1$ and the remaining terms are zero, then the warp function is the identity (i.e., no warp will be applied).

Parameters $(\hat{c}_x, \hat{c}_y)$ are the normalized x- and y-coordinates of the optical center, relative to the top-left pixel of the image.

Example 1: specifying $(0.5, 0.5)$ means that the optical center lies exactly at the image center.

Example 2: specifying $(1, 0)$ means that the optical center lies at the top-right pixel of the image.

Processing of this opcode is performed as follows.

Let $I'_i(x', y')$ be the pixel of the ith plane of the original unwarped image at pixel position (x', y') – i.e., before opcode processing.

Let $I_i(x, y)$ be the pixel value of the ith plane of the warped image at pixel position (x, y) – i.e., after opcode processing.

For each pixel (x, y) of the ith plane of the warped image, compute:

$$I_i(x, y) \xleftarrow[resample]{} I'_i(x', y')$$

i.e., the pixel at position (x', y') in the original unwarped image is effectively moved to position (x, y) in the final warped image, where

$$x' = c_x + m(\Delta x_r + \Delta x_t)$$

$$y' = c_y + m(\Delta y_r + \Delta y_t)$$

$$c_x = x_0 + \hat{c}_x(x_1 - x_0)$$

$$c_y = y_0 + \hat{c}_y(y_1 - y_0)$$

$$m_x = \max(|x_0 - c_x|, |x_1 - c_x|)$$

$$m_y = \max(|y_0 - c_y|, |y_1 - c_y|)$$

$$m = \sqrt{m_x^2 + m_y^2}$$

$$r = \sqrt{\Delta x^2 + \Delta y^2}$$

$$f = k_{r_0,i} + k_{r_1,i}r^2 + k_{r_2,i}r^4 + k_{r_3,i}r^6$$

$$\Delta x = \frac{(x - c_x)}{m}$$

$$\Delta y = \frac{(y - c_y)}{m}$$

$$\Delta x_r = f\Delta x$$

$$\Delta y_r = f\Delta y$$

$$\Delta x_t = k_{t_0,i}(2\Delta x\Delta y) + k_{t_1,i}(r^2 + 2\Delta x^2)$$

$$\Delta y_t = k_{t_1,i}(2\Delta x\Delta y) + k_{t_0,i}(r^2 + 2\Delta y^2)$$

(x_0, y_0) = pixel coordinates of the top-left pixel of the warped image

(x_1, y_1) = pixel coordinates of the bottom-right pixel of the warped image

Notes and Restrictions

$(\Delta x_r, \Delta y_r)$ and $(\Delta x_t, \Delta y_t)$ are the radial and tangential warp components, respectively.

m is the Euclidean distance (in pixels) from the optical center to the farthest pixel in the warped image.

r is the normalized Euclidean distance (in [0, 1]) from the optical center to a given pixel in the warped image.

It is recommended that implementations use a suitable resampling kernel, such as a cubic spline.

This opcode can be used to correct lateral (transverse) chromatic aberration by specifying the appropriate coefficients for each image plane separately.

Each coefficient set K_i must satisfy the following constraints. Let $(x', y') = F(x, y)$ be the 2D warp function defined above. Let $x' = F_x(x, y)$ and $y' = F_y(x, y)$ be the x-component and y-component of $F(x, y)$, respectively. Let $w(r) = k_{r_0,i} r + k_{r_1,i} r^3 + k_{r_2,i} r^5 + k_{r_3,i} r^7$.

The constraints are:

- $F(x, y)$ must be invertible.
- $F_x(x, y)$ must be an increasing function of x for all $x \in [x_0, x_1]$, i.e., $\partial F_x(x, y)/\partial x$.
- $F_y(x, y)$ must be an increasing function of y for all $y \in [y_0, y_1]$, i.e., $\partial F_y(x, y)/\partial y$.
- $w(r)$ must be an increasing function of r for all $r \in [0,1]$, i.e., $w'(r) > 0$.

WarpFisheye

Opcode ID 2

DNG Version 1.3.0.0

Parameters

N LONG

For each coefficient set i $\in 1, 2, \ldots$ N

$k_{r_0,i}$ DOUBLE

$k_{r_1,i}$ DOUBLE

$k_{r_2,i}$ DOUBLE

$k_{r_3,i}$ DOUBLE

$\hat{c}_x$ DOUBLE

$\hat{c}_y$ DOUBLE

Description

This opcode applies a warp to an image and can be used to “unwrap” an image captured with a fisheye lens and map it instead to a perspective projection. It can also be used to correct geometric distortion and lateral (transverse) chromatic aberration for both fisheye and rectilinear lenses.

Let $K_i = \{k_{r_0,i}, k_{r_1,i}, k_{r_2,i}, k_{r_3,i}\}$ denote the ith coefficient set, where $i \in 1, 2, \ldots, N$. Parameter N is the number of coefficient sets. N must be 1 or the total number of image planes. If N = 1, then a single set of warp coefficients (i.e., K_1) is applied to all image planes, i.e., all planes undergo the same transformation. If N > 1, coefficient set K_i is used to process the ith image plane, e.g., K_1 defines the warp function for the first image plane, K_2 defines the warp function for the second image plane, etc.

Parameters $K_i = \{k_{r_0,i}, k_{r_1,i}, k_{r_2,i}, k_{r_3,i}\}$ are the coefficients that define the warp function for the ith image plane; see below for implementation details and restrictions.

Parameters $(\hat{c}_x, \hat{c}_y)$ are the normalized x- and y-coordinates of the optical center, relative to the top-left pixel of the image.

Example 1: specifying $(0.5, 0.5)$ means that the optical center lies exactly at the image center.

Example 2: specifying $(1, 0)$ means that the optical center lies at the top-right pixel of the image.

Processing of this opcode is performed as follows.

Let $I'_i(x', y')$ be the pixel of the ith plane of the original unwarped image at pixel position (x', y') – i.e., before opcode processing.

Let $I_i(x, y)$ be the pixel value of the ith plane of the warped image at pixel position (x, y) – i.e., after opcode processing.

For each pixel (x, y) of the ith plane of the warped image, compute:

$$I_i(x, y) \xleftarrow[resample]{} I'_i(x', y')$$

i.e., the pixel at position (x', y') in the original unwarped image is effectively moved to position (x, y) in the final warped image, where

$$x' = c_x + (m \cdot f \cdot \Delta x)$$

$$y' = c_y + (m \cdot f \cdot \Delta y)$$

$$c_x = x_0 + \hat{c}_x(x_1 - x_0)$$

$$c_y = y_0 + \hat{c}_y(y_1 - y_0)$$

$$m_x = \max(|x_0 - c_x|, |x_1 - c_x|)$$

$$m_y = \max(|y_0 - c_y|, |y_1 - c_y|)$$

$$m = \sqrt{m_x^2 + m_y^2}$$

$$r = \sqrt{\Delta x^2 + \Delta y^2}$$

$$\theta = \arctan(r)$$

$$f = \frac{1}{r}\left(k_{r_0,i}\theta + k_{r_1,i}\theta^3 + k_{r_2,i}\theta^5 + k_{r_3,i}\theta^7\right)$$

$$\Delta x = \frac{(x - c_x)}{m}$$

$$\Delta y = \frac{(y - c_y)}{m}$$

(x_0, y_0) = pixel coordinates of the top-left pixel of the warped image

(x_1, y_1) = pixel coordinates of the bottom-right pixel of the warped image

Notes and Restrictions

m is the Euclidean distance (in pixels) from the optical center to the farthest pixel in the warped image.

r is the normalized Euclidean distance (in [0, 1]) from the optical center to a given pixel in the warped image.

It is recommended that implementations use a suitable resampling kernel, such as a cubic spline.

This opcode can be used to correct lateral (transverse) chromatic aberration by specifying the appropriate coefficients for each image plane separately.

Each coefficient set K_i must satisfy the following constraints.

Let $w(r) = k_{r_0,i}\theta + k_{r_1,i}\theta^3 + k_{r_2,i}\theta^5 + k_{r_3,i}\theta^7$, where $\theta = \arctan(r)$.

The function $w(r)$ must be an increasing function of r for all $r \in [0,1]$, i.e., $w'(r) > 0$.

FixVignetteRadial

Opcode ID 3

DNG Version 1.3.0.0

Parameters

k_0 DOUBLE

k_1 DOUBLE

k_2 DOUBLE

k_3 DOUBLE

k_4 DOUBLE

$\hat{c}_x$ DOUBLE

$\hat{c}_y$ DOUBLE

Description

This opcode applies a gain function to an image and can be used to correct vignetting. Parameters (k_0, k_1, k_2, k_3, k_4) define a radially-symmetric gain function, explained below. Note that if all k_i terms are zero, then the gain function is the identity (i.e., no gain will be applied).

Parameters ($\hat{c}_x$, $\hat{c}_y$) are the normalized x- and y-coordinates of the optical center, relative to the top-left pixel of the image.

Example 1: specifying $(0.5, 0.5)$ means that the optical center lies exactly at the image center.

Example 2: specifying $(1, 0)$ means that the optical center lies at the top-right pixel of the image.

Processing of this opcode is performed as follows.

Let $I'_i(x, y)$ be the uncorrected pixel value of the ith plane of the image at pixel position (x, y), i.e., before opcode processing.

Let $I'(x, y)$ be the corrected pixel value of the ith plane of the image at pixel position (x, y), i.e., after opcode processing.

For each pixel (x, y) in the ith plane of the image, compute:

$$I_i(x, y) = g \cdot I'_i(x, y)$$

where

$$g = 1 + k_0 r^2 + k_1 r^4 + k_2 r^6 + k_3 r^8 + k_4 r^{10}$$

$$c_x = x_0 + \hat{c}_x(x_1 - x_0)$$

$$c_y = y_0 + \hat{c}_y(y_1 - y_0)$$

$$m_x = \max(|x_0 - c_x|, |x_1 - c_x|)$$

$$m_y = \max(|y_0 - c_y|, |y_1 - c_y|)$$

$$m = \sqrt{m_x^2 + m_y^2}$$

$$r = \frac{1}{m}\sqrt{(x - c_x)^2 + (y - c_y)^2}$$

(x_0, y_0) = pixel coordinates of the top-left pixel of the image

(x_1, y_1) = pixel coordinates of the bottom-right pixel of the image

Note that m represents the Euclidean distance (in pixels) from the optical center to the farthest pixel in the image, and r represents the normalized Euclidean distance (in [0, 1]) from the optical center to a given pixel.

FixBadPixelsConstant

Opcode ID 4

DNG Version 1.3.0.0

Parameters

Constant LONG

BayerPhase LONG

Description

This opcode patches (interpolates over) bad pixels in a Bayer pattern CFA image. The bad pixels are marked in the image by setting the bad pixels to a value of Constant.

- If BayerPhase is 0, then the top-left pixel of the image is red pixel.
- If BayerPhase is 1, then the top-left pixel of the image is green pixel in a green/red row.
- If BayerPhase is 2, then the top-left pixel of the image is green pixel in a green/blue row.
- If BayerPhase is 3, then the top-left pixel of the image is blue pixel.

FixBadPixelsList

Opcode ID 5

DNG Version 1.3.0.0

Parameters

BayerPhase LONG

BadPointCount LONG

BadRectCount LONG

For Each BadPointCount:

BadPointRow LONG

BadPointColumn LONG

For Each BadRectCount:

BadRectTop LONG

BadRectLeft LONG

BadRectBottom LONG

BedRectRight LONG

Description

This opcode patches (interpolates over) bad pixels and rectangles in a Bayer pattern CFA image. The bad pixels and rectangles are specified as parameters for this opcode.

- If BayerPhase is 0, then the top-left pixel of the image is red pixel.
- If BayerPhase is 1, then the top-left pixel of the image is green pixel in a green/red row.
- If BayerPhase is 2, then the top-left pixel of the image is green pixel in a green/blue row.
- If BayerPhase is 3, then the top-left pixel of the image is blue pixel.

BadPointCount is the number of bad pixels to patch. BadPointRow and BadPointColumn specify the coordinates of the bad pixels to patch.

BadRectCount is the number of bad rectangles to patch. BadRectTop, BadRectLeft, BadRectBottom, and BadRectRight specify the coordinates of the bad rectangles to patch.

TrimBounds

Opcode ID 6

DNG Version 1.3.0.0

Parameters

Top	LONG
Left	LONG
Bottom	LONG
Right	LONG

Description

This opcode trims the image to the rectangle specified by Top, Left, Bottom, and Right.

MapTable

Opcode ID 7

DNG Version 1.3.0.0

Parameters

Top	LONG
Left	LONG
Bottom	LONG
Right	LONG
Plane	LONG
Planes	LONG
RowPitch	LONG
ColPitch	LONG
TableSize	LONG

For Each TableSize:

TableEntry	SHORT

Description

This opcode maps a specified area and plane range of an image through a 16-bit lookup table (LUT).

The bounds of the area of the image to be affected is specified by the Top, Left, Bottom, Right parameters. The first plane, and the number of planes, to be modified are specified by the Plane and Planes parameters. If RowPitch not equal to one, then only every RowPitch rows starting at the Top are affected. If ColPitch is not equal to one, then only every ColPitch columns starting at Left are affected.

TableSize specifies the number of entries in the 16-bit LUT. TableEntry specifies each table entry.

MapPolynomial

Opcode ID 8

DNG Version 1.3.0.0

Parameters

Top	LONG
Left	LONG
Bottom	LONG
Right	LONG
Plane	LONG
Planes	LONG
RowPitch	LONG
ColPitch	LONG
Degree	LONG

For Each Value 0…Degree:

Coefficient	DOUBLE

Description

This opcode maps a specified area and plane range of an image through a polynomial function.

The bounds of the area of the image to be affected is specified by the Top, Left, Bottom, Right parameters. The first plane, and the number of planes, to be modified are specified by the Plane and Planes parameters. If RowPitch not equal to one, then only every RowPitch rows starting at the Top are affected. If ColPitch is not equal to one, then only every ColPitch columns starting at Left are affected.

The mapping function is a polynomial of degree Degree. The maximum allowed value for Degree is 8. The coefficients are stored in increasing order, starting with the zero degree coefficient (constant term).

GainMap

Opcode ID 9

DNG Version 1.3.0.0

Parameters

Top	LONG
Left	LONG
Bottom	LONG
Right	LONG
Plane	LONG
Planes	LONG
RowPitch	LONG
ColPitch	LONG
MapPointsV	LONG
MapPointsH	LONG
MapSpacingV	DOUBLE
MapSpacingH	DOUBLE
MapOriginV	DOUBLE
MapOriginH	DOUBLE
MapPlanes	LONG

For Each MapPointsV:

For Each MapPointsH:

For Each MapPlanes:

MapGain FLOAT

Description

This opcode multiplies a specified area and plane range of an image by a gain map.

The bounds of the area of the image to be affected is specified by the Top, Left, Bottom, Right parameters. The first plane, and the number of planes, to be modified are specified by the Plane and Planes parameters. If RowPitch not equal to one, then only every RowPitch rows starting at the Top are affected. If ColPitch is not equal to one, then only every ColPitch columns starting at Left are affected.

The gain map is a sub-sampled 2-D 32-bit floating-point image. MapPointsV is the number of samples in vertical direction. MapPointsH is the number of samples in the horizontal direction.

The gain map is not required to cover the entire image being modified. Inside the gain map bounds, values are interpolated using bi-linear interpolation. Outside the gain map bounds, values are replicated from the edges of the gain map.

The origin of the gain map relative to image being modified is specified by MapOriginV (vertical direction) and MapOriginH (horizontal direction), which are in relative coordinates, where 1.0 is equal to the height or width of the image being modified, respectively.

The spacing between gain map points is specified by MapSpacingV (vertical direction) and MapSpacingH (horizontal direction). Again these are in relative coordinates, where 1.0 is equal to the height or width of the image being modified, respectively.

MapPlanes specifies the number of planes in the gain map. If Planes > MapPlanes, the last gain map plane is used for any remaining planes being modified.

DeltaPerRow

Opcode ID 10

DNG Version 1.3.0.0

Parameters

Top	LONG
Left	LONG
Bottom	LONG
Right	LONG
Plane	LONG
Planes	LONG
RowPitch	LONG
ColPitch	LONG
Count	LONG

For Each Count:

Delta	FLOAT

Description

This opcode applies a per-row delta (constant offset) to a specified area and plane range of an image.

The bounds of the area of the image to be affected is specified by the Top, Left, Bottom, Right parameters. The first plane, and the number of planes, to be modified are specified by the Plane and Planes parameters. If RowPitch not equal to one, then only every RowPitch rows starting at the Top are affected. If ColPitch is not equal to one, then only every ColPitch columns starting at Left are affected.

Count is the number of deltas, and is required to match the number of affected rows in the specified area.

The delta to add to each affected row is specified by Delta.

DeltaPerColumn

Opcode ID 11

DNG Version 1.3.0.0

Parameters

Top	LONG
Left	LONG
Bottom	LONG
Right	LONG
Plane	LONG
Planes	LONG
RowPitch	LONG
ColPitch	LONG
Count	LONG

For Each Count:

Delta	FLOAT

Description

This opcode applies a per-column delta (constant offset) to a specified area and plane range of an image.

The bounds of the area of the image to be affected is specified by the Top, Left, Bottom, Right parameters. The first plane, and the number of planes, to be modified are specified by the Plane and Planes parameters. If RowPitch not equal to one, then only every RowPitch rows starting at the Top are affected. If ColPitch is not equal to one, then only every ColPitch columns starting at Left are affected.

Count is the number of deltas, and is required to match the number of affected columns in the specified area.

The delta to add to each affected column is specified by Delta.

ScalePerRow

Opcode ID 12

DNG Version 1.3.0.0

Parameters

Top	LONG
Left	LONG
Bottom	LONG
Right	LONG
Plane	LONG
Planes	LONG
RowPitch	LONG
ColPitch	LONG
Count	LONG

For Each Count:

Scale	FLOAT

Description

This opcode applies a per-row scale to a specified area and plane range of an image.

The bounds of the area of the image to be affected is specified by the Top, Left, Bottom, Right parameters. The first plane, and the number of planes, to be modified are specified by the Plane and Planes parameters. If RowPitch not equal to one, then only every RowPitch rows starting at the Top are affected. If ColPitch is not equal to one, then only every ColPitch columns starting at Left are affected.

Count is the number of scale values, and is required to match the number of affected rows in the specified area.

The scale to multiply each affected row by is specified by Scale.

ScalePerColumn

Opcode ID 13

DNG Version 1.3.0.0

Parameters

Top	LONG
Left	LONG
Bottom	LONG
Right	LONG
Plane	LONG
Planes	LONG
RowPitch	LONG
ColPitch	LONG
Count	LONG

For Each Count:

Scale	FLOAT

Description

This opcode applies a per-column scale to a specified area and plane range of an image.

The bounds of the area of the image to be affected is specified by the Top, Left, Bottom, Right parameters. The first plane, and the number of planes, to be modified are specified by the Plane and Planes parameters. If RowPitch not equal to one, then only every RowPitch rows starting at the Top are affected. If ColPitch is not equal to one, then only every ColPitch columns starting at Left are affected.

Count is the number of scale values, and is required to match the number of affected columns in the specified area.

The scale to multiply each affected column by is specified by Scale.

WarpRectilinear2

Opcode ID 14

DNG Version 1.6.0.0

Parameters

N	LONG

For each coefficient set $i \in 1, 2, \ldots N$

$k_{r_0,i}$	DOUBLE
$k_{r_1,i}$	DOUBLE
$k_{r_2,i}$	DOUBLE
…	
$k_{r_{14},i}$	DOUBLE
$k_{t_0,i}$	DOUBLE
$k_{t_1,i}$	DOUBLE
min_valid_radius	DOUBLE
max_valid_radius	DOUBLE
$\hat{c}_x$	DOUBLE
$\hat{c}_y$	DOUBLE
reciprocalRadial	LONG

Description

This opcode is an extension of the existing WarpRectilinear opcode introduced in DNG 1.3. The goal is to support the correction of more complex (higher-order) distortion. The changes are:

1. Additional radial polynomial terms. The original WarpRectilinear opcode supports radial distortion as a sum of even-power radial terms, up to r^6 order. This opcode supports both odd- and even-power radial terms up to r^{14} order.
2. An option to apply the radial distortion model using division.
3. An option to specify a valid radius range for the radial polynomial model.

In this opcode, the radial distortion term f is computed as:

$$f(r) = \sum_{p=0}^{14} \left(k_{r_{p,i}} \cdot r^p \right)$$

If the reciprocalRadial parameter is 0, then the definition of the radial distortion term is:

$$r_{real} = r_{ideal} \cdot f\left(\text{clamp}(r_{ideal}, \text{ min_valid_radius}, \text{ max_valid_radius})\right)$$

If the reciprocalRadial parameter is a non-zero value, then the definition of the radial distortion term is:

$$r_{real} = \frac{r_{ideal}}{f\left(\text{clamp}(r_{ideal},\ \text{min_valid_radius},\ \text{max_valid_radius})\right)}$$

The **min_valid_radius** and **max_valid_radius** parameters define a valid range for the radial polynomial function f. This may be useful if optimizing a warp function to match another lens's field of view.

Additional details:

- The above function f must be one-to-one for values of r_{ideal} in the range [min_valid_radius, max_valid_radius].
- When r_{ideal} < min_valid_radius, then the input to f is clamped to min_valid_radius; see the above definition of the radial distortion term.
- When r_{ideal} > max_valid_radius, then the input to f is clamped to max_valid_radius; see the above definition of the radial distortion term.
- For normal use cases (where the radial distortion model is valid over the entire recorded image area), set min_valid_radius to 0.0 and max_valid_radius to 1.0.
- These parameters must satisfy the inequalities: $0.0 \leq$ min_valid_radius < max_valid_radius ≤ 1.0.

The meaning of the other parameters is the same as WarpRectilinear.

Special Compatibility Note and Relationship to Previous Warp Opcodes

This section describes a modification to the normal DNG opcode processing model. This modification is intended to maximize warp support and compatibility across DNG readers. The idea is that a DNG opcode list can use two warp opcodes: a more accurate opcode (WarpRectilinear2) intended for DNG 1.6 and later readers, and a baseline opcode (either WarpRectilinear or WarpFisheye) for earlier DNG readers.

Normally, a DNG reader applies all the opcodes in an opcode list in sequence from front to back. However, if the reader processes a WarpRectilinear2 opcode that is marked **optional** (using the opcode flag bits), then the reader should skip the next opcode in the list if it's a WarpRectilinear or WarpFisheye opcode. This "skip" rule applies only to the opcode immediately following the WarpRectilinear2 opcode, if any, in the same opcode list.

Example 1: Opcode list 3 contains the following opcodes.

1. WarpRectilinear2
2. GainMap

Result: The reader should apply both opcodes, in order. The opcode following WarpRectilinear2 is neither WarpRectilinear nor WarpFisheye, so it is applied normally.

Example 2: Opcode list 3 contains the following opcodes:

1. WarpRectilinear2, marked optional
2. WarpFisheye
3. GainMap

Result: The reader should apply the WarpRectilinear2 opcode, then the GainMap opcode. The WarpFisheye opcode is skipped, per the skip rule.

Example 3: Opcode list 3 contains the following opcodes:

1. WarpRectilinear2, marked optional
2. GainMap
3. WarpRectilinear

Result: The reader should apply all three opcodes, in order. Although there is a WarpRectilinear opcode later in the list, it does not immediately follow WarpRectilinear2, so the skip rule does not apply.

Appendix A: Compatibility With Previous Versions

This appendix documents only the differences between this and previous versions of the DNG specification that are relevant to compatibility. Differences that are not relevant to compatibility (e.g., new optional tags that DNG readers are not required to support) are not documented in this appendix.

This information is useful in enabling DNG readers to correctly read DNG files with older version numbers. It also helps determine what version DNG writers can include in the DNGBackwardVersion tag.

The first version of the DNG speciation that was published was version 1.0.0.0.

Compatibility Issue 1: ActiveArea Tag

The ActiveArea tag was added to the DNG specification in version 1.1.0.0. Previous versions of the DNG specification do not support storing masked pixels.

DNG writers should set the DNGBackwardVersion to a minimum of 1.1.0.0 if the masked pixels are stored in the DNG file.

Compatibility Issue 2: 16-bit Lossless JPEG Encoding

The Lossless JPEG encoder/decoder used by Adobe applications to read and write DNG files before version 1.1.0.0 incorrectly deviated from the JPEG specification when dealing with 16- bit data. Since both the encoder and decoder deviated in the same way, no data was lost; however the data stream did not exactly match the data stream specified in the Lossless JPEG specification.

Because the vast majority of DNG 1.0.0.0 files using 16-bit Lossless JPEG encoding were created by Adobe applications, it is strongly recommended that software that reads or writes DNG files with version numbers less than 1.1.0.0 incorporate this deviation. Software that reads or writes DNG files with version 1.1.0.0 or later can safely assume that the Lossless JPEG stream is fully compliant with the Lossless JPEG specification.

Description of deviation

Lossless JPEG encodes the difference between a predicted value and the actual value for each pixel. With 16-bit data, these differences are computed modulo 16-bits, so the range of possible differences is -32768 to + 32767. Two values are stored for the difference. First the number of bits required to store the difference (encoded via a Huffman code), and then the actual difference.

Using the difference encoding scheme in the Lossless JPEG specification, only one difference value would take 16-bits to store: -32768. The Lossless JPEG specification special cases this difference bit length, and since there is only one possible difference value it does not bother to use any bits to store the actual difference.

In earlier versions of DNG the special case logic is not present, and the difference value for - 32768 is stored in the compressed data stream as with all other difference bit lengths.

Compatibility Issue 3: SubTileBlockSize

The SubTileBlockSize tag was added to the DNG specification in version 1.2.0.0. Earlier versions of the DNG specification did not support this feature.

DNG writers should set the DNGBackwardVersion tag to a minimum of 1.2.0.0 if non-default values are used for this tag.

Compatibility Issue 4: RowInterleaveFactor

The RowInterleaveFactor tag was added to the DNG specification in version 1.2.0.0. Earlier versions of the DNG specification did not support this feature.

DNG writers should set the DNGBackwardVersion tag to a minimum of 1.2.0.0 if non-default values are used for this tag.

Compatibility Issue 5: PreviewColorSpace

The PreviewColorSpace tag was added to the DNG specification in version 1.2.0.0. Earlier versions of the DNG specification did not support this feature.

DNG writers should set the DNGBackwardVersion tag to a minimum of 1.2.0.0 if non-default values are used for this tag.

Compatibility Issue 6: CFALayout

CFALayout values 6 through 9 were added to the DNG specification in version 1.3.0.0. Earlier versions of the DNG specification did not support these CFALayout values.

DNG writers should set the DNGBackwardVersion tag to a minimum of 1.3.0.0 if CFALayout values 6 through 9 are used.

Compatibility Issue 7: Opcode Lists

The OpcodeList1, OpcodeList2, and OpcodeList3 tags, plus the initial set of supported Opcode IDs, were added to the DNG specification in version 1.3.0.0.

DNG writers should not set the DNGBackwardVersion to a version less than the DNG version for any opcode that is not marked optional.

Compatibility Issue 8: Floating Point Image Data

Support for floating point image data was added in DNG Version 1.4.0.0.

DNG writers should set the DNGBackwardVersion tag to a minimum of 1.4.0.0 if the main image IFD uses floating point data.

Compatibility Issue 9: Transparent Pixels

Support for transparent pixels was added in DNG Version 1.4.0.0.

DNG writers should set the DNGBackwardVersion tag to a minimum of 1.4.0.0 if the DNG file contains an IFD with NewSubFileType equal to 4.

Compatibility Issue 10: Deflate Compression

Support for compression code 8 (Deflate) was added in DNG Version 1.4.0.0.

DNG writers should set the DNGBackwardVersion tag to a minimum of 1.4.0.0 if the main image IFD uses compression code 8.

Compatibility Issue 11: Lossy JPEG Compression

Support for compression code 34892 (Lossy JPEG) was added in DNG Version 1.4.0.0.

DNG writers should set the DNGBackwardVersion tag to a minimum of 1.4.0.0 if the main image IFD uses compression code 34892.

Compatibility Issue 12: Depth Maps

Support for depth maps was added in DNG Version 1.5.0.0, but since reading the depth map is optional, writers should not increase the DNGBackwardVersion tag higher than otherwise required.

Compatibility Issue 13: Enhanced Image Data

Support for enhanced image data was added in DNG Version 1.5.0.0, but since reading the enhanced image data is optional, writers should not increase the DNGBackwardVersion tag higher than otherwise required.

Compatibility Issue 14: Semantic Mask

Support for semantic masks was added in DNG Version 1.6.0.0, but since reading the semantic mask is optional, writers should not increase the DNGBackwardVersion tag higher than otherwise required.

Compatibility Issue 15: ProfileGainTableMap

Support for ProfileGainTableMap was added in DNG Version 1.6.0.0, but since reading and applying the ProfileGainTableMap is optional, writers should not increase the DNGBackwardVersion tag higher than otherwise required.

Compatibility Issue 16: WarpRectilinear2 Opcode

Support for the WarpRectilinear2 opcode was added in DNG Version 1.6.0.0. DNG writers should set the DNGBackwardVersion as described in Compatibility Issue 7: Opcode Lists. Namely, if a DNG writer includes a WarpRectilinear2 opcode that is marked as required, then the DNGBackwardVersion tag must be set to at least 1.6.0.0. Also see the description of WarpRectilinear2 in Chapter 7 for a method of combining WarpRectilinear2 with WarpRectilinear or WarpFisheye to maximize compatibility across DNG readers.

Compatibility Issue 17: Third Color Calibration and Custom Illuminant Data

Support for a third color calibration was added in DNG Version 1.6.0.0. To maximize compatibility across readers, a DNG writer is required to set the DNGBackwardVersion tag to at least 1.6.0.0 only if at least one of the following conditions is met:

- CalibrationIlluminant1 tag is present and set to 255
- CalibrationIlluminant2 tag is present and set to 255

Compatibility Issue 18: JPEG XL Compression

Support for compression code 52546 (JPEG XL) was added in DNG Version 1.7.0.0.

DNG writers should set the DNGBackwardVersion tag to a minimum of 1.7.0.0 if the main image IFD uses compression code 52546.

Compatibility Issue 19: ColumnInterleaveFactor

The ColumnInterleaveFactor tag was added to the DNG specification in version 1.7.1.0. Earlier versions of the DNG specification did not support this feature.

DNG writers should set the DNGBackwardVersion tag to a minimum of 1.7.1.0 if non-default values are used for this tag.

Compatibility Issue 20: ProfileGainTableMap and Camera Profile IFD

Support for ProfileGainTableMap in Camera Profile IFD was added in DNG Version 1.7.0.0, but since reading and applying the ProfileGainTableMap is optional, writers should not increase the DNGBackwardVersion tag higher than otherwise required.