Windows Media™ Photo

Photographic Still Image File Format

Feature Specification

Version	1.0
Status	Final Draft

Microsoft Corporation Technical Documentation License Agreement for the specification "Windows Media™ Photo"

READ THIS! THIS IS A LEGAL AGREEMENT BETWEEN MICROSOFT CORPORATION ("MICROSOFT") AND THE RECIPIENT OF THE ABOVE REFERENCED MATERIALS, WHETHER AN INDIVIDUAL OR AN ENTITY ("YOU"). IF YOU HAVE ACCESSED THIS AGREEMENT IN THE PROCESS OF DOWNLOADING THESE MATERIALS ("MATERIALS") FROM A MICROSOFT WEB SITE, BY CLICKING "I ACCEPT", DOWNLOADING, USING OR PROVIDING FEEDBACK ON THE MATERIALS, YOU AGREE TO THESE TERMS. IF THIS AGREEMENT IS ATTACHED TO MATERIALS, BY ACCESSING, USING OR PROVIDING FEEDBACK ON THE ATTACHED MATERIALS, YOU AGREE TO THESE TERMS. IF YOU DO NOT AGREE TO THESE TERMS, YOU ARE NOT AUTHORIZED TO ACCESS, DOWNLOAD, USE OR REVIEW THE MATERIALS.

For good and valuable consideration, the receipt and sufficiency of which are acknowledged, You and Microsoft agree as follows:

1. You may review these Materials only (a) as a reference to assist You in planning and designing Your product, service or technology ("Product") to interface with a Microsoft product, specification, service or technology ("Microsoft Product") as described in these Materials; and (b) to provide feedback on these Materials to Microsoft. All other rights are retained by Microsoft; this Agreement does not give You rights under any Microsoft patents. You may not (i) duplicate any part of these Materials, (ii) remove this Agreement or any notices from these Materials, or (iii) give any part of these Materials, or assign or otherwise provide Your rights under this Agreement, to anyone else.

2. These Materials may contain preliminary information or inaccuracies, and may not correctly represent any associated Microsoft Product as commercially released. All Materials are provided entirely "AS IS." To the extent permitted by law, MICROSOFT MAKES NO WARRANTY OF ANY KIND, DISCLAIMS ALL EXPRESS, IMPLIED AND STATUTORY WARRANTIES, AND ASSUMES NO LIABILITY TO YOU FOR ANY DAMAGES OF ANY TYPE IN CONNECTION WITH THESE MATERIALS OR ANY INTELLECTUAL PROPERTY IN THEM.

3. If You are an entity and (a) merge into another entity or (b) a controlling ownership interest in You changes, Your right to use these Materials automatically terminates and You must destroy them.

4. You have no obligation to give Microsoft any suggestions, comments or other feedback ("Feedback") relating to these Materials. However, any Feedback you voluntarily provide may be used in Microsoft Products and related specifications or other documentation (collectively, "Microsoft Offerings") which in turn may be relied upon by other third parties to develop their own products, services or technology ("Third Party Products"). Accordingly, if You do give Microsoft Feedback on any version of these Materials or the Microsoft Offerings to which they apply, You agree: (a) Microsoft may freely use, reproduce, license, distribute, and otherwise commercialize Your Feedback in any Microsoft Offering; (b) You also grant third parties, without charge, only those patent rights necessary to enable Third Party Products to use, implement or interface with any specific parts of a Microsoft Product that incorporate Your Feedback; and (c) You will not give Microsoft any Feedback (i) that You have reason to believe is subject to any patent, copyright or other intellectual property claim or right of any third party; or (ii) subject to license terms which seek to require any Microsoft Offering incorporating or derived from such Feedback, or other Microsoft intellectual property, to be licensed to or otherwise shared with any third party.

5. Microsoft has no obligation to maintain the confidentiality of any Microsoft Offering, or the confidentiality of Your Feedback, including Your identity as the source of such Feedback.

6. This Agreement is governed by the laws of the State of Washington. Any dispute involving it must be brought in the federal or state superior courts located in King County, Washington, and You waive any defenses allowing the dispute to be litigated elsewhere. If there is litigation, the losing party must pay the other party's reasonable attorneys' fees, costs and other expenses. If any part of this Agreement is unenforceable, it will be considered modified to the extent necessary to make it enforceable, and the remainder shall continue in effect. This Agreement is the entire agreement between You and Microsoft concerning these Materials; it may be changed only by a written document signed by both You and Microsoft.

Contents

Preface

About This Specification

Windows Media™ Photo is a file format and associated codec specifically designed to for use with all types of continuous tone photographic content. This document describes the features and capabilities of the Windows Media Photo Version 1.0 release. This version is compatible with the implementation of Windows Media Photo in the released versions of Windows Vista, Windows Presentation Foundation (WPF) and Windows Imaging Component (WIC).

The information contained in this specification is subject to change. Every effort has been made to ensure accuracy at the time of publication.

This specification is written for developers who are implementing support for Windows Media Photo in applications, including support for the XML Paper Specification (XPS).

Chapters 1 through 4 contain information about the file format itself; Chapter 5 contains information specific for those developing applications for Windows Vista and other supported Windows versions (such as Windows XP) using the WPF or WIC runtime components.

Licensing Notes

Certain information relating to Windows Media Photo, including the details of the image compression algorithm is available only to licensees of the technology.

The bit stream level documentation of the Windows Media Photo compression technology is documented in the Windows Media™ Photo Device Porting Kit (DPK). Information on licensing this DPK, including for use in XPS, is available at www.microsoft.com/windows/windowsmedia/wmphoto.

Trademark Notice

Windows Media™ is a registered trademark of Microsoft Corporation. All rights are reserved.

Formatting Conventions

This specification uses the following formatting conventions:

Terms are formatted like *this*.

Important comments, typically highlighting unimplemented or preliminary features look like this.

```
Code looks like this.
```

Raw text and editorial notes look like this.

Language Notes

In this specification, the words that are used to define the significance of each particular requirement are capitalized. These words are used in accordance with their definitions in RFC 2119 and their meaning is reproduced here for convenience:

- MUST. This word, or the adjective "REQUIRED," means that the item is an absolute requirement of the specification.
- SHOULD. This word, or the adjective "RECOMMENDED," means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
- MAY. This word, or the adjective "OPTIONAL," means that this item is truly optional. For example, one implementation may choose to include the item because a particular marketplace or scenario requires it or because it enhances the product. Another implementation may omit the same item.

Chapter 1. OVERVIEW

1.1 Objectives for Introducing a New Still Image Format

Today's file formats for continuous tone images present many limitations in maintaining the highest image quality or delivering the most optimal system performance. Windows Media Photo was designed to remove these limitations. The design objectives include:

- High performance, embedded system friendly compression
    - Small memory footprint
    - Simple, integer-only operations (no divides)
- Industry-leading compression quality
- Lossless or lossy compression using the same algorithm
- Support a very wide range of pixel formats:
    - Monochrome, RGB, CMYK or n-Channel image representation
    - 8 or 16-bit unsigned integer
    - 16 or 32-bit signed integer
    - 16 or 32-bit floating point
    - Several packed bit formats
        - 1bpc monochrome
        - 5 or 10bpc RGB
        - RGBE Radiance
- Simple, extensible TIFF-like container structure
- Planar or interleaved alpha channel
- Embedded ICC Profile
- EXIF and XMP metadata

Windows Media Photo is the only format that offers high dynamic range image encoding, lossless or lossy compression, multiple color formats, and performance that enables practical in-device implementation.

1.2 Compression Algorithm Overview

Windows Media Photo employs a new, state-of-the-art compression algorithm optimized for the digital photography market. Windows Media Photo offers image quality comparable to JPEG-2000 with computational and memory performance more closely comparable to JPEG. Windows Media Photo delivers a lossy compressed image of better perceptive quality than JPEG at less than half the file size. The same compression algorithm can also deliver mathematically lossless compressed images that are typically 2.5 times smaller than the original uncompressed data.

Windows Media Photo uses a very high performance reversible color space conversion, a reversible lapped biorthogonal transform and an advanced non-arithmetic entropy coding scheme. The combination of these new technologies offers extremely high compression efficiency with minimal loss of important image content. Windows Media Photo typically surpasses other lossy image compression technologies in preserving high frequency detail while simultaneously minimizing objectionable spatial artifacts.

The compression algorithm used in Windows Media Photo is computationally efficient, and is designed for high performance encoding and decoding while minimizing system resource requirements. The core compression transform requires at most 3 non-trivial (multiply plus addition) and 7 trivial (addition or shift) operations per pixel (with no divisions) at the highest quality level. In the highest performance mode, only 1 non-trivial and 4 trivial operations per pixel are required. The image is processed in 16x16 macro blocks, allowing a minimal memory footprint for embedded implementations.

Windows Media Photo provides native support for both RGB and CMYK, providing a reversible color transform for each of these color formats to an internal luminance-dominant format used for optimal compression efficiency. In addition Windows Media Photo supports monochrome and arbitrary n-channel color formats.

Because the transforms employed are fully reversible, the codec supports both lossless and lossy operation using a single algorithm. This significantly simplifies the implementation for embedded applications and provides capabilities not typically found in other compressed image file formats.

Windows Media Photo supports a wide range of popular numerical encodings at multiple bit depths. 8-bit and 16-bit formats, as well as some specialized packed bit formats, are supported for both lossy and lossless compression. 32-bit formats are only supported using lossy compression as only 24 bits are typically retained through the various transforms designed to achieve maximum compression efficiency. While Windows Media Photo uses integer arithmetic exclusively for its internal processing, an innovative color transform process provides lossless encoding support for both fixed and floating point image information. This also enables extremely efficient conversion between different color formats as part of the encode/decode process.

The technical details of the Windows Media Photo compression algorithm are documented in the Windows Media Photo Device Porting Kit (see Preface.)

1.3 Format Identity

The file extension for a Windows Media Photo file is **WDP**. The MIME type for a Windows Media Photo file is **image/vnd.ms-photo**.

Chapter 2. Image Data Encoding

2.1 Overview

Windows Media Photo supports a wide range of color encoding formats including monochrome, RGB, CMYK and n-channel colors using several different fixed and floating point numerical representations at multiple bit depths, providing support for a very wide range of data compression scenarios.

The overall goal is to support the greatest possible level of image dynamic range and color precision, maintain forward compatibility with existing formats, and keep the device implementations of the encoder and decoder as simple as possible. To that end, the formats supported by Windows Media Photo are divided into *Basic* and *Advanced* formats. The minimum requirements for a decoder for digital photography applications include support for the *Basic* formats. The *Advanced* formats can optionally be supported by decoders targeted for other application-specific scenarios including printing, 3D rendering or advanced imagery applications. An encoder need only support the specific formats required for its particular scenarios. For example, a digital camera encoder has no need to support CMYK color or bit depths or channel configurations beyond the capabilities of the camera's sensor. A general purpose image application should ideally support all the formats supported by Windows Media Photo. Since the underlying components in Windows provide this support, this is a simple requirement for any Windows application.

Windows Media Photo does not directly support palletized (indexed) color formats. However, these formats can readily be converted to one of the formats directly supported by Windows Media Photo.

The table listing pixel formats supported by Windows Media Photo, including whether these are *Basic* or *Advanced* formats, is included in the metadata section of this document, under the PixelFormat tag.

2.2 Numerical Formats

The numerical format includes the bit depth and the numerical encoding. The bit depth specifies how many bits of data are used to describe each value. The numerical encoding specifies how Windows Media Photo translates the range of values at a specific bit depth into a numerical value.

Today, most digital photography scenarios use a bit depth of 8; each of the channels that together describe an image pixel is represented by 8 bits, providing 256 unique values. For more demanding applications, it is not uncommon to use a bit depth of 16, providing 65536 unique values to describe each channel within a pixel. In some less common scenarios, even greater bit depths are used. In scenarios when memory or processing power is at a premium, as few as five bits may be used, providing only 32 unique values per channel.

Regarding numerical encoding, most common photo and image formats use an 8-bit or 16-bit unsigned integer value to represent the intensity of each color channel. The minimum value (0) represents zero intensity in a single channel. Black is achieved when all channels are zero. A maximum value represents 100% intensity or full saturation of an individual channel. When all channels are maximal, this corresponds to white

The exact meaning of "full saturation" and "white" as well as the specific colors produced by all the intermediate numerical steps in the unsigned integer range is dependent both on how these values are initially created or captured and how these values are rendered. Different source or destination devices (including cameras, scanners, displays or printers) may produce different numerical values to represent the same "real world" color.

While it might be possible to agree on one method for assigning specific numerical values to real world colors, doing so is problematic. Since any specific device has its own limited range for color reproduction, the resulting overlap between the agreed-upon universal color range and the device's range may be a small portion of the large spectrum of desired "real world" colors. As a result, such an approach is an extremely inefficient use of the available numerical values, especially when using only 8 bits (or 256 unique values) per channel.

To represent pixel values as efficiently as possible, devices use a numeric encoding optimized for their own range of possible colors or *gamut*. In addition, a *color profile* is provided that describes the numeric encoding for the specific device relative to some pre-defined reference standard. This color profile includes the specification of the (typically) non-linear transformation from the range of integer values to a uniform set of "brightness" steps based on the appearance of the resulting displayed color value. This non-linear transformation is often referred to as the gamma curve or *gamma* of the color profile. The gamma is often simplified to a single numerical value specifying this transformation as a power function. A color profile makes it possible to convert image information between different color profiles, thereby improving the ability to produce the same "real world" colors from a variety of devices.

To maintain full compatibility with existing legacy devices and applications, Windows Media Photo supports a variety of color profiled pixel formats using unsigned integer representations in bit depths of 8 and 16, as well as smaller bit depths for specialized applications. Additionally, Windows Media Photo also supports a number of advanced pixel formats that avoid many of the limitations and complexity imposed by unsigned integer representations.

The process of limiting the numerical representation of colors to the range specified by a particular device creates some significant restrictions. First, any color value that cannot be displayed within the gamut of the chosen color profile is discarded, even though that color may be displayable if the image data is converted to a different color profile in the future.

In addition, any intermediate image processing has the potential to produce values that extend beyond the black (zero saturation) or white (maximum saturation) point of the particular color profile, resulting in these calculated values to be clipped to the associated limits. Thus, the limited numerical range caused these values to be corrupted during intermediate calculations, even though subsequent image processing may bring these values back within the displayable (black to white) range of the target rendering destination. Because of this issue, most modern image processing software uses a much larger gamut, represented by numerical values of greater bit depths, for all intermediate image processing calculations. (It is not uncommon to use 32-bit floating point values for intermediate calculations, minimizing any image corruption caused by rounding errors and clipping at the range limits during the intermediate steps of image processing calculations.) However, most common image formats today require that this image data be converted back to a range-limited, unsigned integer representation, limited to the gamut as defined by a specific color profile. So once again, the potential for data corruption exists.

To address these challenges, Windows Media Photo provides a much more flexible approach to the numerical encoding of image data by supporting a wide range of different pixel formats. Windows Media Photo supports three types of numerical encoding, each at a variety of bit depths.

2.2.1 Unsigned Integer

Windows Media Photo supports both 8-bit and 16-bit unsigned integers in a variety of pixel formats. Windows Media Photo also supports a few "packed bit" formats that all used unsigned integer representations at other bit depths. In all cases, a value of zero represents minimum saturation or black for the specific channel and the maximum possible value represents the maximum viewable value for that channel. When all viewable channels for a pixel format are at their maximum numerical value, this corresponds to the brightest viewable color, or white.

For 8-bit unsigned integer, the maximum value is 255, providing 256 unique values. For 16-bit unsigned integer, the maximum value is 65535, providing 65536 unique values. This document will also use the abbreviation UINT to refer to unsigned integer values.

2.2.2 Fixed Point

A fixed point numerical representation is not common in today's image file formats. It is being introduced in Windows Media Photo as an optimal format to encode greater dynamic ranges while still retaining all the performance advantages of integer processing.

Fixed point values are essentially signed, scaled integer values. By applying an appropriate scaling factor, the signed integer range can represent an arbitrary numeric range. This enables the encoding of color information that goes beyond the traditional range limits of "black" and "white" or the gamut of any particular device or rendering target.

Rather than interpreting the number as range of integer steps from black to maximum saturation for a particular color profile, a fixed point number is scaled to represent a floating point value in an unprofiled color space. In this unprofiled space, zero still represents the minimum visible value, or black. A value of 1.0 represents the maximum visible value, or when applied to all channels that make up a pixel, white. The specific scaling for each bit depth specifies exactly what point in the entire signed integer range is interpreted as a value of 1.0.

Additionally, unlike unsigned integer values that are almost always interpreted based on a gamma curve that is optimized for the particular color profile, this unprofiled space is based on a linear transformation (or a gamma equal to 1.0)

Windows Media Photo supports fixed point numerical encoding for 16-bit and 32-bit signed values. This document will also use the abbreviation SINT to refer to signed integer, or fixed point values.

2.2.2.1 16-bit Fixed Point – s2.13

The 16 bits that make up an individual value are interpreted as a sign bit, two integer bits and thirteen fractional bits. The shorthand description for this encoding is s2.13.

Using this interpretation, an unprofiled numerical range of -4.0 to +3.999+ can be represented, and the "white" value of 1.0 is represented by the signed integer value 8,192 (0x2000h).

2.2.2.2 32-bit Fixed Point – s7.24

The 32 bits that make up an individual value are interpreted as a sign bit, seven integer bits and twenty-four fractional bits. The shorthand description for this encoding is s7.24.

Using this interpretation, an unprofiled numerical range of -128.0 to +127.999+ can be represented, and the "white" value of 1.0 is represented by the signed integer value 16,777,216 (0x01000000h).

Windows Media Photo does not provide 100% lossless compression for 32-bit data. The encoding and decoding algorithms use 32-bit computations, so some precision is lost during these calculations. A minimum of 22 bits and typically 24 bits or more precision is retained through the end-to-end encoding and decoding process.

2.2.3 Floating Point

For some applications, the dynamic range provided by a fixed point representation may not be sufficient. Therefore, Windows Media Photo also supports a floating point numerical representation. Floating point will not compress as efficiently, but it provides a dramatically wider dynamic range than fixed point.

Similar to fixed point, floating point values represent image intensity within a linear, unprofiled space. In this unprofiled space, zero still represents the minimum visible value, or black. A floating point value of 1.0 represents the maximum visible value, or when applied to all channels that make up a pixel, white.

Windows Media Photo supports floating point numerical encoding for 16-bit and 32-bit depths. A special packed bit RGB float format is also supported.

2.2.3.1 16-bit Floating Point – HALF s5e10

The 16 bits are formatted in accordance with the HALF floating point format, with one sign bit, 5 exponent bits and 10 normalized mantissa bits. This provides an efficient method to encode values with a very wide dynamic range. The short hand description for this encoding is s5e10.

2.2.3.2 32-bit Floating Point – IEEE s8e23

The numerical value is encoded in accordance with the 32-bit implementation of the ANSI/IEEE Standard 754-1985 *Standard for Binary Floating Point Arithmetic*, widely used on most computing platforms. The format uses one sign bit, 8 exponent bits and 23 normalized mantissa bits. While this is one of the least efficient means to encode values, it offers the greatest precision and dynamic range. The short hand description for this encoding is s8e23.

As noted above, Windows Media Photo does not provide 100% lossless compression for 32-bit data. The encoding and decoding algorithms use 32-bit computations, so some precision is lost during these calculations. A minimum of 22 bits and typically 24 bits or more precision is retained through the end-to-end encoding and decoding process.

2.2.3.3 32bpp (16bpc) RGB Shared Exponent Floating Point

This special packed bit representation encodes three 16-bit floating point values using four bytes. The bytes include unnormalized, unsigned 8-bit mantissas for the red, green and blue channels, plus a shared 8-bit exponent. While this offers no increase in gamut, it is a more compact uncompressed method to encode image content with a very wide exposure range. Windows Media Photo supports compression of this representation without the need to first convert to a different format.

2.3 Channel Organizations

Some of the more popular formats for digital photos only support the encoding of three-channel, red/green/blue (RGB) content. Windows Media Photo supports a variety of different channel structures and organizations, providing considerably more flexibility for encoding image information.

2.3.1 RGB and BGR

By far the most common method of describing an image is with three separate channels, representing the additive primary colors red, green and blue. To best support legacy pixel formats, Windows Media Photo interprets these three channels in either red-green-blue or blue-green-red sequential order, as specified by the appropriate pixel format identifier.

It is very important--especially for 8 bit per channel (bpc) formats--that the pixel format nomenclature refers to the order of the color channels within the sequential bit stream describing the uncompressed bitmap. Historically, RGB has commonly been used to refer to the order of the 8bpc channel values when stored within a 32bit word representing one pixel. However, on a little-endian system, the byte ordering within the 32-bit word is opposite of the actual byte order within the sequential bit stream. For 8bpc, 16bpc and 32bpc pixel formats, Windows Media Photo channel order nomenclature ALWAYS refers to the channel order within the sequential bit stream.

The exception to the channel order nomenclature as described above is with any packed bit formats. In this case, the channel values span byte boundaries, so the channel order name refers to the order of the channel information within the 16-bit or 32-bit word that describes the pixel.

Windows Media Photo does not support BGR and RGB ordering for every numerical encoding. In general, BGR channel order is used for 8bpc content and RGB order is used for other bit depths. For legacy reasons, 8bpc is supported in both BGR and RGB channel order.

Here is the list of RGB and BGR pixel formats:

PixelFormat	Ch	BPC	BPP	Num
24bppRGB	3	8	24	UINT
24bppBGR	3	8	24	UINT
32bppBGR	3	8	24	UINT
48bppRGB	3	16	48	UINT
48bppRGBFixedPoint	3	16	48	SINT
48bppRGBHalf	3	16	48	Float
96bppRGBFixedPoint	3	32	96	SINT
128bppRGBFloat	3	32	128	Float
16bppBGR555	3	5	16	UINT
16bppBGR565	3	5,6,5	16	UINT
32bppBGR101010	3	10	32	UINT
32bppRGBE	3	16	32	Float

2.3.2 RGBA and BGRA

For of the RGB or BGR pixel formats, Windows Media Photo supports a corresponding pixel format that adds an alpha channel. Here is the list of RGB/BGR pixel formats that include alpha:

PixelFormat	Ch	BPC	BPP	Num
32bppBGRA	4	8	32	UINT
64bppRGBA	4	16	64	UINT
64bppRGBAFixedPoint	4	16	64	SINT
64bppRGBAHalf	4	16	64	Float
128bppRGBAFixedPoint	4	32	128	SINT
128bppRGBAFloat	4	32	128	Float

2.3.3 PRGBA and PBGRA

Windows Media Photo also supports a subset of formats that include pre-multiplied alpha channels. In these formats, the stored red, green and blue data values have already been multiplied by the alpha channel value. Pre-multiplication is useful because it makes alpha-based image compositing more efficient. If an application needs the original RGB values, the alpha channel multiplication step must be reversed.

Windows Media Photo provides support for pre-multiplied alpha channel RGB content in the following pixel formats:

PixelFormat	Ch	BPC	BPP	Num
32bppPBGRA	4	8	32	UINT
64bppPRGBA	4	16	64	UINT
128bppPRGBAFloat	4	32	128	Float

2.3.4 Gray

Monochrome or gray scale image content can be described using a single channel. Windows Media Photo supports the following pixel formats for monochrome images:

PixelFormat	Ch	BPC	BPP	Num
8bppGray	1	8	8	UINT
16bppGray	1	16	16	UINT
16bppGrayFixedPoint	1	16	16	SINT
16bppGrayHalf	1	16	16	Float
32bppGrayFixedPoint	1	32	32	SINT
32bppGrayFloat	1	32	32	Float
BlackWhite	1	1	1	UINT

2.3.5 CMYK

The most common generic method to describe images for printing is using four channels to represent cyan, magenta, yellow and black. Windows Media Photo supports both 8bpc and 16bpc unsigned integer pixel formats for CMYK:

PixelFormat	Ch	BPC	BPP	Num
32bppCMYK	4	8	32	UINT
64bppCMYK	4	16	64	UINT

2.3.6 CMYKA

Windows Media Photo also supports the inclusion of an alpha channel with CMYK data in both 8bpc and 16bpc unsigned integer pixel formats:

PixelFormat	Ch	BPC	BPP	Num
40bppCMYKAlpha	5	8	40	UINT
80bppCMYKAlpha	5	16	80	UINT

2.3.7 n-Channel

To provide maximum flexibility, Windows Media Photo allows the encoding of image data in no predefined channel organization. From three to eight channels of continuous tone data may be encoded in either 8bpc or 16bpc unsigned integer format. n-Channel encoding is less efficient than using a pre-defined channel organization because the Windows Media Photo encoder can not make any assumptions about the correlation among the channels. However, this encoding allows for a wide variety of image data formats. n-Channel encoding is most commonly used for printing applications where it is desirable to store image content in the target color space of a multi-ink printer.

Windows Media Photo supports the following n-channel pixel formats:

PixelFormat	*Ch*	*BPC*	*BPP*	*Num*
24bpp3Channels	3	8	24	UINT
32bpp4Channels	4	8	32	UINT
40bpp5Channels	5	8	40	UINT
48bpp6Channels	6	8	48	UINT
56bpp7Channels	7	8	56	UINT
64bpp8Channels	8	8	64	UINT
48bpp3Channels	3	16	48	UINT
64bpp4Channels	4	16	64	UINT
80bpp5Channels	5	16	80	UINT
96bpp6Channels	6	16	96	UINT
112bpp7Channels	7	16	112	UINT
128bpp8Channels	8	16	128	UINT

2.3.8 n-Channel with Alpha

Windows Media Photo also supports the following pixel formats that are equivalent to the preceding 8bpc and 16bpc n-Channel pixel formats plus an alpha channel:

PixelFormat	*Ch*	*BPC*	*BPP*	*Num*
32bpp3ChannelsAlpha	4	8	32	UINT
40bpp4ChannelsAlpha	5	8	40	UINT
48bpp5ChannelsAlpha	6	8	48	UINT
56bpp6ChannelsAlpha	7	8	56	UINT
64bpp7ChannelsAlpha	8	8	64	UINT
72bpp8ChannelsAlpha	9	8	72	UINT
64bpp3ChannelsAlpha	4	16	64	UINT
80bpp4ChannelsAlpha	5	16	80	UINT
96bpp5ChannelsAlpha	6	16	96	UINT
112bpp6ChannelsAlpha	7	16	112	UINT
128bpp7ChannelsAlpha	8	16	128	UINT
144bpp8ChannelsAlpha	9	16	144	UINT

2.4 Color Context

The color context for a Windows Media Photo image can be defined explicitly, based on an embedded ICC color profile, an EXIF color format tag, or, implicitly, based on predefined default interpretations for each pixel format.

2.4.1 ICC Profiles

An embedded ICC profile provides a non-ambiguous definition of an image's color context and is an ideal solution for certain pixel formats. By definition, ICC profiles only define the unsigned visible gamut. Therefore, ICC profiles are suitable for use with unsigned integer pixel formats but inappropriate for fixed point or floating point pixel formats.

Fixed point or floating point pixel formats should always be encoded in an unprofiled, linear (gamma = 1.0) color space. If an ICC profile is included with an image using a fixed point or floating point pixel format, the profile does not describe the interpretation of the numerical values. It is only used to specify the desired target profile if and when the image data is converted to an unsigned integer pixel format.

2.4.2 EXIF ColorSpace Metadata Tag (0xA000)

If the EXIF tag is present, applications may choose to use this information to define the color context of the image. However, this is an optional specification and should be ignored if an ICC profile is present. The EXIF 2.2 specification only defines values for *sRGB* (1) and *Uncalibrated* (0xFFFF) for this tag. All other values are reserved.

As previously noted, fixed point or floating point pixel formats should always be encoded in an unprofiled, linear (gamma = 1.0) color space. If the EXIF ColorSpace tag is included with an image using a fixed point or floating point pixel format, the profile does not describe the interpretation of the numerical values. It is only used to specify the desired target profile if and when the image data is converted to an unsigned integer pixel format.

2.4.3 Default Color Context

In the absence of any metadata to describe the color context, the following defaults should be assumed for Windows Media Photo files, based on the pixel format.

2.4.3.1 Unsigned Integer RGB

In the absence of an ICC profile, unsigned integer RGB data is assumed to use the sRGB color space (www.color.org/sRGB.html). If the EXIF ColorSpace tag is present, it either defines the color context as *sRGB* or *Uncalibrated*. Unsigned integer data that is tagged as *Uncalibrated* can be assumed to use the sRGB color space. Therefore, the EXIF ColorSpace tag can be effectively ignored for unsigned integer RGB image content, unless it is used for some non-standard, application specific purpose.

2.4.3.2 Fixed or floating point RGB

Any fixed or floating point RGB data should be encoded using the scRGB color space. scRGB as used in a Windows Media Photo file is an unprofiled, linear (gamma = 1.0) color space that uses the color primaries and illuminant as defined by sRGB. The desired visible black point is specified by the numerical value 0.0 and the desired visible white point is specified by all three color channels set to a value of 1.0.

Fixed or floating point numerical encoding allows the description of image content outside the visible range, and therefore cannot be correctly described using an ICC Profile. If either an ICC profile or EXIF ColorSpace tag is present, this does not affect the encoding of the fixed or floating point image content; it specifies the desired target color space if and when the image is converted to an unsigned integer format. It may also specify the limits of the image content's color gamut.

2.4.3.3 Unsigned Integer Gray

In the absence of an ICC profile, unsigned integer Gray content is assumed to use a color space equivalent to sRGB with all three channels sharing the same value.

2.4.3.4 Fixed or floating point Gray

Fixed or floating point Gray content should be encoded using an unprofiled, linear (gamma = 1.0) color space with a value of 0.0 representing the desired visible black point and 1.0 representing the desired visible white point.

As described above, fixed or floating point numerical encoding allows the description of image content outside the visible range, and therefore cannot be correctly described using an ICC Profile. If an ICC profile is present, this does not affect the encoding of the fixed or floating point image content; it specifies the desired target color space if and when the image is converted to an unsigned integer format

2.4.3.5 CMYK

In the absence of an ICC profile, any CMYK pixel format is assumed to be encoded using the SWOP (Specifications for Web Offset Publications) color space (www.swop.org/specification).

2.4.3.6 n-Channel

There is no inherent description of the color context for n-Channel data, so when using an n-Channel pixel format, an ICC profile should always be included. However, if an ICC profile is not present, the following default color context assumptions apply

- n = 3 the three channels are assumed to be red, green, and blue (RGB) encoded using the sRGB color space
- n > 3 the first four channels are assumed to be cyan, yellow, magenta and black (CMYK) encoded using the SWOP color space. Any additional channels are ignored.

Chapter 3. Windows Media Photo Container

3.1 IFD Container

Windows Media Photo stores image data in a container organized as a table of Image File Directory (IFD) tags, similar to a TIFF 6.0 container. A standard Windows Media Photo file contains one or more images using individual, linked IFD tables. Each image contains the following elements:

- Image data
- Optional planar alpha channel
- Basic Windows Media Photo metadata stored as IFD tags
- Optional descriptive metadata stored as IFD tags
- Optional XMP metadata encoded in XML and stored as a single IFD tag with extended data
- Optional EXIF metadata stored as a sub IFD table linked by an IFD tag
- Optional ICC color profile stored as an IFD tag with extended data.

The image data is a monolithic self contained, self describing Windows Media Photo compressed data structure.

A planar alpha channel is stored as separately compressed single channel image data, referenced by the appropriate IFD tags. This enables the image to be decoded independently of the alpha channel.

In an effort to remain compatible with software designed to decode IFD-table based TIFF files, the largest possible Windows Media Photo file is $2^{32}-1$ bytes in length. This limit will be addressed in a future update.

All multi-byte numerical values in a Windows Media Photo file are stored in "little-endian" format, starting with the least significant bytes in the serial byte stream. Windows Media Photo does not support the internal use of "big-endian" encoding for pointers, metadata values or other internal data elements. Supporting only one endian encoding limits the additional work of dealing with the two different formats to only those systems or devices which are natively big-endian, rather than requiring every decoder implementation, regardless of the native format to accommodate both possible encodings.

> That said, it is perfectly reasonable to implement a Windows Media Photo codec on little-endian architecture systems. The Device Porting Kit provides reference source code to support both types of system architectures.

A Windows Media Photo file begins with an 8-byte file header that points to an image file directory (IFD). An image file directory contains information about the photo, as well as pointers to the actual photo data.

Figure 1 and the following paragraphs describe the Windows Media Photo file header and IFD in more detail.

3.2 Windows Media Photo File Header

A Windows Media Photo file begins with an 8-byte photo file header (see Figure 1), containing the following information:

Bytes 0-1: "II" (0x4949) - This corresponds to the TIFF header convention for little-endian byte order for multi-byte numerical formats. Windows Media Photo only supports little-endian encoding, so the first two bytes of the file will always be "II"

Byte 2: 0xBC – a unique byte value (188) that identifies the file as a Windows Media Photo file vs. a TIFF 6.0 or other TIFF style file. While it would be preferable to have a longer and more deterministic identification field, the restrictions of maintaining compatibility with TIFF header format restricts this to a single byte.

Byte 3: The version number of the Windows Media Photo file structure. At present, the only allowable version numbers are 0 and 1. 0 is reserved to represent pre-release development versions of the bit stream. A version 0 file may contain data incompatible with the final version of the format. A Windows Media Photo file that fully conforms to the released 1.0 Windows Media Photo specification must always have a version number of 1. Values greater than 1 are reserved for future versions of the file format. Since the interpretation of any information beyond the first four bytes of the file may change in future versions, a 1.0 compatible decoder must reject any files with a version number greater than 1.

Bytes 4-7: The offset (in bytes) of the first IFD. The directory may be at any location in the file after the header but must begin on a word boundary. In particular, an IFD may follow the image data it describes. Readers must follow the pointers wherever they may lead.

The term byte offset is always used in this document to refer to a location with respect to the beginning of the Windows Media Photo file. The first byte of the file has an offset of 0.

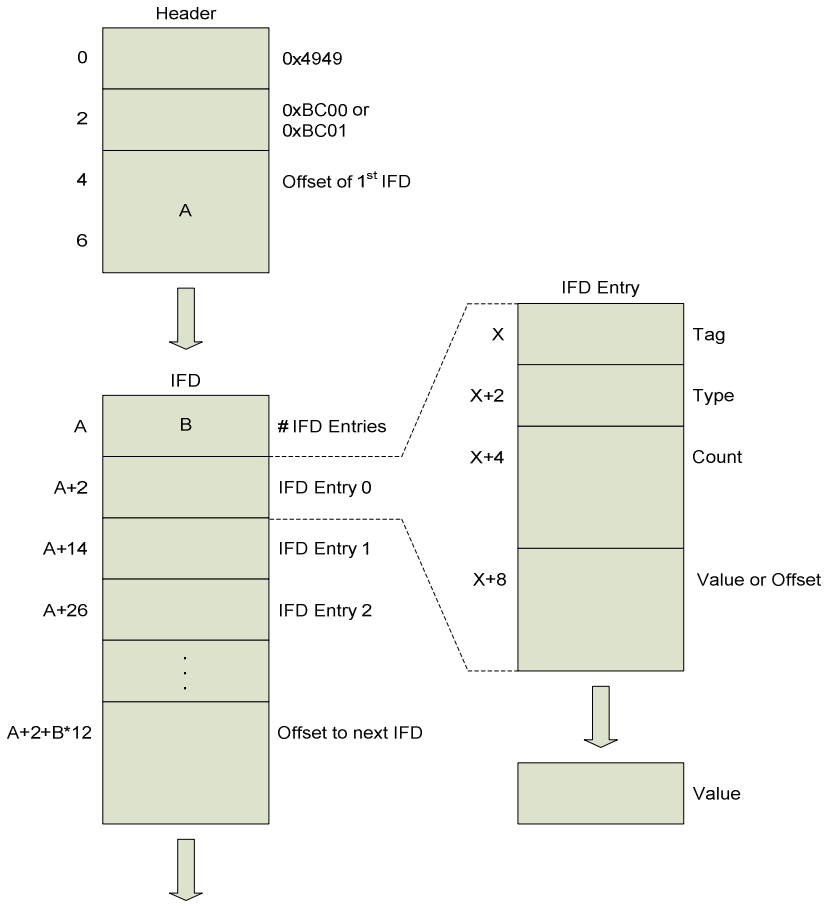

Figure 1 – Windows Media Photo File Header and IFD Table Structure

3.3 Image file directory

An Image file directory (IFD) consists of a 2-byte count of the number of directory entries (i.e., the number of fields), followed by a sequence of 12-byte field entries, and followed by a 4-byte byte offset to the next IFD (or 0 if none). (Do not forget to write the 4 bytes of 0 after the last IFD.) There must be at least 1 IFD in a Windows Media Photo file and each IFD must have at least one entry. See Figure 1.

3.3.1 IFD Entry

Each 12-byte IFD entry has the following format:

Bytes 0-1 The *Tag* that identifies the field.

Bytes 2-3 The field *Type*.

Bytes 4-7 The number of values, *Count* of the indicated Type.

Bytes 8-11 The *Value/Offset*, contains the tag value if it is four bytes (or less) or the file offset (in bytes) of the location in the file of the Value for this tag. The Value is expected to begin on a word boundary; the corresponding Value Offset will thus be an even number. This file offset may point anywhere in the file, even after the photo data.

3.3.2 Sort Order

The entries in an IFD must be sorted in ascending order by Tag. Note that this is not the order in which the fields are described in this document. When an IFD entry contains an Offset that points to additional data, these additional data elements may be stored in the file in any order.

3.3.3 Value/Offset

To save time and space the Value/Offset contains the Value instead of pointing to the Value if and only if the Value fits into 4 bytes. If the Value is shorter than 4 bytes, it is left-justified within the 4-byte Value/Offset, i.e., stored in the lower numbered bytes. (If this value is read into a 32-bit register on a little-endian machine, this will correspond the least significant bytes.) Whether the Value fits within 4 bytes is determined by the Type and Count of the field.

3.3.4 Count

Count is the number of values. Note that Count is not the total number of bytes. For example, a single 16-bit word (SHORT) has a Count of 1; not 2.

3.3.5 Types

The field types and their sizes are:

1 = BYTE	8-bit unsigned integer.
2 = ASCII	8-bit byte that contains a 7-bit ASCII code; the last byte must be NUL (binary zero).
3 = SHORT	16-bit (2-byte) unsigned integer in little-endian (LSB first) byte order.
4 = LONG	32-bit (4-byte) unsigned integer in little-endian (LSB first) byte order.
5 = RATIONAL	Two LONGs: the first represents the numerator of a fraction; the second, the denominator, both in little-endian (LSB first) byte order.
6 = SBYTE	An 8-bit signed (twos-complement) integer.
7 = UNDEFINED	An 8-bit byte that may contain anything, depending on the definition of the field.
8 = SSHORT	A 16-bit (2-byte) signed (twos-complement) integer in little-endian (LSB first) byte order.
9 = SLONG	A 32-bit (4-byte) signed (twos-complement) integer in little-endian (LSB first) byte order.
10 = SRATIONAL	Two SLONG's: the first represents the numerator of a fraction, the second the denominator, both in little-endian (LSB first) byte order.
11 = FLOAT	Single precision (4-byte) IEEE format in little-endian (LSB first) byte order.
12 = DOUBLE	Double precision (8-byte) IEEE format in little-endian (LSB first) byte order.

The value of the Count part of an ASCII field entry includes the NUL. If padding is necessary, the Count does not include the pad byte. Note that there is no initial "count byte" as in Pascal-style strings.

Any ASCII field can contain multiple strings, each terminated with a NUL. A single string is preferred whenever possible. The Count for multi-string fields is the number of bytes in all the strings in that field

plus their terminating NUL bytes. Only one NUL is allowed between strings, so that the strings following the first string will often begin on an odd byte.

The reader must check the type to verify that it contains an expected value. Windows Media Photo currently allows more than 1 valid type for some fields. For example, ImageWidth and ImageLength are usually specified as having type SHORT. But photos with more than 64K rows or columns must use the LONG field type.

Windows Media Photo readers must accept BYTE, SHORT, or LONG values for any unsigned integer field. This allows a single procedure to retrieve any integer value, makes reading more robust, and saves disk space in some situations.

> Warning: It is possible that other Windows Media Photo field types will be added in the future. Readers must not try to interpret fields containing an unexpected field type.

3.3.6 Fields are arrays

Each IFD entry has an associated Count. This means that all fields are actually one-dimensional arrays, even though most fields contain only a single value. For example, to store a complicated data structure in a single private field, use the UNDEFINED field type and set the Count to the number of bytes required to hold the data structure.

3.4 Multiple Images per Windows Media Photo File

There may be more than one IFD in a Windows Media Photo file. The default view of the Windows Media Photo file is always stored as the first frame so a basic Windows Media Photo reader is not required to read any IFD's beyond the first one.

Windows Media Photo allows the storage of an alternate view of an image, typically implemented as a thumbnail or a reduced resolution preview. This is stored as an additional image within the file and identified by the appropriate metadata tags.

Chapter 4. Windows Media Photo Tags

4.1 Image Format

Rather than use a series of metadata tags to attempt to describe the attributes of an image's structure, Windows Media Photo uses a unique GUID to provide a non-ambiguous definition of the image pixel format. Each pixel format value represents a unique definition of all the parameters that describe the image format. An encoder or decoder maintains a list of the GUID's it supports with a table of the associated pixel attributes. This eliminates any ambiguity over unsupported combinations of individual image attribute tags.

4.1.1 PixelFormat

Tag = 48129 (0xBC01)

Type = BYTE

Count = 16

A 128-bit Globally Unique Identifier (GUID) that identifies the image pixel format. This is a required metadata tag with no default value.

The tables starting on the next page list all the pixel formats currently supported by the Windows Media Photo codec. Each table contains the following information:

PixelFormat Name	A descriptive identification for the purpose of this document. The Windows Avalon WIC symbol for the GUID is this name, preceded by "GUID_".
GUID	The globally unique identifier that specifies this pixel format.
Ch	The number of channels.
BPC	The number of bits per channel.
BPP	The number of bits per pixel. In the case when this value is not equal to Ch * BPP there are additional padding bits
Num	The numerical interpretation of the value, either an unsigned integer, a signed integer or a floating point number. Signed integers are always in two's complement format. 32-bit floating point numbers are in IEEE format. 16-bit floating point numbers are in Half format.
Color	The basic color structure of the image. Windows Media Photo supports image data structured as single channel monochrome (Gray), three-channel RGB, four-channel CMYK or n-Channel data containing anywhere from two to sixteen channels of arbitrary, uncorrelated continuous tone information. The detailed information that describes the colorimetric attributes of the image is stored in the ICC color profile.
A	Indicates that this format includes an alpha channel.
B	Indicates that this is a *Basic* pixel format supported by all Windows Media Photo decoder implementations.

4.1.1.1 RGB

PixelFormat Name GUID	Ch	BPC	BPP	Num	Color	A	B
WICPixelFormat24bppRGB **6fddc324-4e03-4bfe-b1853d77768dc90d**	3	8	24	UINT	RGB		✓
WICPixelFormat24bppBGR **6fddc324-4e03-4bfe-b1853d77768dc90c**	3	8	24	UINT	RGB		✓
WICPixelFormat32bppBGR **6fddc324-4e03-4bfe-b1853d77768dc90e**	3	8	24	UINT	RGB		
WICPixelFormat48bppRGB **6fddc324-4e03-4bfe-b1853d77768dc915**	3	16	48	UINT	RGB		✓
WICPixelFormat48bppRGBFixedPoint **6fddc324-4e03-4bfe-b1853d77768dc912**	3	16	48	SINT	RGB		✓
WICPixelFormat48bppRGBHalf **6fddc324-4e03-4bfe-b1853d77768dc93b**	3	16	48	Float	RGB		
WICPixelFormat96bppRGBFixedPoint **6fddc324-4e03-4bfe-b1853d77768dc918**	3	32	96	Float	RGB		
WICPixelFormat128bppRGBFloat **6fddc324-4e03-4bfe-b1853d77768dc91b**	3	32	128	Float	RGB		
WICPixelFormat32bppBGRA **6fddc324-4e03-4bfe-b1853d77768dc90f**	4	8	32	UINT	RGB	✓	
WICPixelFormat64bppRGBA **6fddc324-4e03-4bfe-b1853d77768dc916**	4	16	64	UINT	RGB	✓	
WICPixelFormat64bppRGBAFixedPoint **6fddc324-4e03-4bfe-b1853d77768dc91d**	4	16	64	SINT	RGB	✓	
WICPixelFormat64bppRGBAHalf **6fddc324-4e03-4bfe-b1853d77768dc93a**	4	16	64	Float	RGB	✓	
WICPixelFormat128bppRGBAFixedPoint **6fddc324-4e03-4bfe-b1853d77768dc91e**	4	32	128	SINT	RGB	✓	
WICPixelFormat128bppRGBAFloat **6fddc324-4e03-4bfe-b1853d77768dc919**	4	32	128	Float	RGB	✓	
WICPixelFormat32bppPBGRA **6fddc324-4e03-4bfe-b1853d77768dc910**	4	8	32	UINT	RGB	✓	
WICPixelFormat64bppPRGBA **6fddc324-4e03-4bfe-b1853d77768dc917**	4	16	64	UINT	RGB	✓	
WICPixelFormat128bppPRGBAFloat **6fddc324-4e03-4bfe-b1853d77768dc91a**	4	32	128	Float	RGB	✓	

4.1.1.2 CMYK

PixelFormat Name GUID	Ch	BPC	BPP	Num	Color	A	B
WICPixelFormat32bppCMYK **6fddc324-4e03-4bfe-b1853d77768dc91c**	4	8	32	UINT	CMYK		
WICPixelFormat40bppCMYKAlpha **6fddc324-4e03-4bfe-b1853d77768dc92c**	5	8	40	UINT	CMYK	✓	
WICPixelFormat64bppCMYK **6fddc324-4e03-4bfe-b1853d77768dc91f**	4	16	64	UINT	CMYK		
WICPixelFormat80bppCMYKAlpha **6fddc324-4e03-4bfe-b1853d77768dc92d**	5	16	80	UINT	CMYK	✓	

4.1.1.3 n-Channel

PixelFormat Name GUID	Ch	BPC	BPP	Num	Color	A	B
WICPixelFormat24bpp3Channels **6fddc324-4e03-4bfe-b1853d77768dc920**	3	8	24	UINT	N-Chn		
WICPixelFormat32bpp4Channels **6fddc324-4e03-4bfe-b1853d77768dc921**	4	8	32	UINT	N-Chn		
WICPixelFormat40bpp5Channels **6fddc324-4e03-4bfe-b1853d77768dc922**	5	8	40	UINT	N-Chn		
WICPixelFormat48bpp6Channels **6fddc324-4e03-4bfe-b1853d77768dc923**	6	8	48	UINT	N-Chn		
WICPixelFormat56bpp7Channels **6fddc324-4e03-4bfe-b1853d77768dc924**	7	8	56	UINT	N-Chn		
WICPixelFormat64bpp8Channels **6fddc324-4e03-4bfe-b1853d77768dc925**	8	8	64	UINT	N-Chn		
WICPixelFormat32bpp3ChannelsAlpha **6fddc324-4e03-4bfe-b1853d77768dc92e**	4	8	32	UINT	N-Chn	✓	
WICPixelFormat40bpp4ChannelsAlpha **6fddc324-4e03-4bfe-b1853d77768dc92f**	5	8	40	UINT	N-Chn	✓	
WICPixelFormat48bpp5ChannelsAlpha **6fddc324-4e03-4bfe-b1853d77768dc930**	6	8	48	UINT	N-Chn	✓	
WICPixelFormat56bpp6ChannelsAlpha **6fddc324-4e03-4bfe-b1853d77768dc931**	7	8	56	UINT	N-Chn	✓	
WICPixelFormat64bpp7ChannelsAlpha **6fddc324-4e03-4bfe-b1853d77768dc932**	8	8	64	UINT	N-Chn	✓	
WICPixelFormat72bpp8ChannelsAlpha **6fddc324-4e03-4bfe-b1853d77768dc933**	9	8	72	UINT	N-Chn	✓	
WICPixelFormat48bpp3Channels **6fddc324-4e03-4bfe-b1853d77768dc926**	3	16	48	UINT	N-Chn		
WICPixelFormat64bpp4Channels **6fddc324-4e03-4bfe-b1853d77768dc927**	4	16	64	UINT	N-Chn		
WICPixelFormat80bpp5Channels **6fddc324-4e03-4bfe-b1853d77768dc928**	5	16	80	UINT	N-Chn		
WICPixelFormat96bpp6Channels **6fddc324-4e03-4bfe-b1853d77768dc929**	6	16	96	UINT	N-Chn		
WICPixelFormat112bpp7Channels **6fddc324-4e03-4bfe-b1853d77768dc92a**	7	16	112	UINT	N-Chn		
WICPixelFormat128bpp8Channels **6fddc324-4e03-4bfe-b1853d77768dc92b**	8	16	128	UINT	N-Chn		
WICPixelFormat64bpp3ChannelsAlpha **6fddc324-4e03-4bfe-b1853d77768dc934**	4	16	64	UINT	N-Chn	✓	
WICPixelFormat80bpp4ChannelsAlpha **6fddc324-4e03-4bfe-b1853d77768dc935**	5	16	80	UINT	N-Chn	✓	
WICPixelFormat96bpp5ChannelsAlpha **6fddc324-4e03-4bfe-b1853d77768dc936**	6	16	96	UINT	N-Chn	✓	
WICPixelFormat112bpp6ChannelsAlpha **6fddc324-4e03-4bfe-b1853d77768dc937**	7	16	112	UINT	N-Chn	✓	
WICPixelFormat128bpp7ChannelsAlpha **6fddc324-4e03-4bfe-b1853d77768dc938**	8	16	128	UINT	N-Chn	✓	
WICPixelFormat144bpp8ChannelsAlpha **6fddc324-4e03-4bfe-b1853d77768dc939**	9	16	144	UINT	N-Chn	✓	

4.1.1.4 Gray

PixelFormat Name GUID	Ch	BPC	BPP	Num	Color	A	B
WICPixelFormat8bppGray **6fddc324-4e03-4bfe-b1853d77768dc908**	1	8	8	UINT	Gray		✓
WICPixelFormat16bppGray **6fddc324-4e03-4bfe-b1853d77768dc90b**	1	16	16	UINT	Gray		✓
WICPixelFormat16bppGrayFixedPoint **6fddc324-4e03-4bfe-b1853d77768dc913**	1	16	16	SINT	Gray		✓
WICPixelFormat16bppGrayHalf **6fddc324-4e03-4bfe-b1853d77768dc93e**	1	16	16	Float	Gray		
WICPixelFormat32bppGrayFixedPoint **6fddc324-4e03-4bfe-b1853d77768dc93f**	1	32	32	SINT	Gray		
WICPixelFormat32bppGrayFloat **6fddc324-4e03-4bfe-b1853d77768dc911**	1	32	32	Float	Gray		

4.1.1.5 Packed Bits

PixelFormat Name GUID	Ch	BPC	BPP	Num	Color	A	B
WICPixelFormatBlackWhite **6fddc324-4e03-4bfe-b1853d77768dc905**	1	1	1	UINT	Gray		
WICPixelFormat16bppBGR555 **6fddc324-4e03-4bfe-b1853d77768dc909**	3	5	16	UINT	RGB		
WICPixelFormat16bppBGR565 **6fddc324-4e03-4bfe-b1853d77768dc90a**	3	5,6,5	16	UINT	RGB		
WICPixelFormat32bppBGR101010 **6fddc324-4e03-4bfe-b1853d77768dc913**	3	10	32	UINT	RGB		
WICPixelFormat32bppRGBE **6fddc324-4e03-4bfe-b1853d77768dc93d**	3	16	32	Float	RGB		

4.2 Rows and Columns

An image is organized as a rectangular array of pixels. The dimensions of this array are described by the fields listed in the following subsections. If an application directly changes the *Transformation* metadata tag (effectively requesting a transform to be performed when the image is decoded), then the application must also make sure the values for ImageWidth, ImageHeight, WidthResolution and HeightResolution are correct, and make the appropriate swap between width and height values if the resulting transformation change includes a rotation. For Windows applications written using the WIC interfaces, it will never be necessary to directly access these metadata tags. All information in these tags are queried and changed through WIC interfaces.

4.2.1 ImageWidth

Tag = 48256 (0hBC80)

Type = LONG

Count = 1

This specifies the number of columns in the transformed photo, or the number of pixels per scan line. Because an ill-behaved application can potentially modify this value so that it does not represent the true image width, applications should call the appropriate codec programmable interface to query image dimensions whenever possible. ImageWidth is a required metadata tag with no default value.

4.2.2 ImageHeight

Tag = 48257 (0hBC81)

Type = LONG

Count = 1

This specifies the number of rows of pixels or scan lines in the transformed photo. Whenever possible, applications should call the appropriate codec programmable interface to query the image dimensions rather than rely on the value of this metadata tag. ImageHeight is a required metadata tag with no default value.

4.2.3 WidthResolution

Tag = 48258 (0hBC82)

Type = FLOAT

N = 1

This optional tag specifies the horizontal resolution of a transformed image expressed in pixels per inch. If this value is zero or this optional tag is not present, then a default resolution of 96dpi is assumed.

4.2.4 HeightResolution

Tag = 48259 (0hBC83)

Type = FLOAT

Count = 1

This optional tag specifies the vertical resolution of a transformed image expressed in pixels per inch. If this value is zero or this optional tag is not present then a default resolution of 96dpi is assumed.

4.3 Location of the Data

Because Windows Media Photo uses an advanced compression scheme, there is no simple way for applications to directly access specific portions of the stored photo data other than through the appropriate codec interfaces. Windows Media Photo files include the tags described in the following subsections to specify the location and size of the compressed image data structure as well as the optional compressed alpha channel data structure.

Windows Applications written using the WIC APIs will never need to directly access these metadata tags. WIC interfaces provide a more convenient and safer method for all operations that make use of these metadata values.

4.3.1 ImageOffset

Tag = 48320 (0hBCC0)

Type = LONG

Count = 1

This specifies the byte offset pointer to the beginning of the photo data, relative to the beginning of the file. This is a required metadata tag with no default value.

4.3.2 ImageByteCount

Tag = 48321 (0hBCC1)

Type = LONG

Count = 1

This specifies the size of the photo data in bytes. This is a required metadata tag with no default value.

4.3.3 AlphaOffset

Tag = 48322 (0hBCC2)

Type = LONG

Count = 1

If this optional metadata tag is present and non-zero, it specifies the byte offset pointer the beginning of the planar alpha channel data, relative to the beginning of the file. If this metadata tag is present then the AlphaByteCount tag must also be present. If the image contains an interleave-encoded alpha channel, then the planar alpha channel is ignored.

4.3.4 AlphaByteCount

Tag = 48323 (0hBCC3)

Type = LONG

Count = 1

If this optional metadata tag is present and non-zero, it specifies the size of the alpha channel data in bytes. If this metadata tag is present, then the AlphaOffset tag must also be present. If the image contains an interleave-encoded alpha channel, then the planar alpha channel is ignored.

4.3.5 Uncompressed

Tag = 48131 (0hBC03)

Type = LONG

Count = 1

This tag specifies if the image data is uncompressed. If this optional metadata tag is not present or zero, this specifies that the image is compressed using the Windows Media Photo compression algorithm. A value of 1 specifies an uncompressed image.

Uncompressed mode is reserved for future use and is not currently implemented or supported.

4.3.6 Transformation

Tag = 48130 (0xBC02)

Type = LONG

Count = 1

This optional metadata tag specifies the transformation to be applied when decoding the image to present the desired representation.

There are 8 different possible image orientations as the result of the combination of a 90 degree clockwise rotation, a horizontal flip and a vertical flip. The Transformation tag represents the required transformation to achieve each orientation, as shown in the following table. ID=0 represents the un-transformed image:

ID	*RCW*	*FlipH*	*FlipV*	*Orient*	*Fill*	*TIFF*
0	0	0	0		TL	1
1	0	0	1		BL	4
2	0	1	0		TR	2
3	0	1	1		BR	3
4	1	0	0		RT	6
5	1	0	1		RB	7
6	1	1	0		LT	5
7	1	1	1		LB	8

The columns in the preceding table are as follows:

ID The tag value that specifies the transformation.

RCW A value of one specifies that a 90-degree clockwise rotation is applied as the first step of the transformation.

FlipV A value of one specifies that vertical flip is applied as part of the transformation, following the rotation.

FlipH A value of one specifies that horizontal flip is applied as part of the transformation, following the rotation.

Orient This graphically shows the resulting orientation as a result of the transformation.

Fill This is an alternate way to describe the requested transformation, specifying the associated two-dimensional fill order of the bitmap as follows:

TL The 0^{th} row represents the top edge of the image and the 0^{th} column represents the left edge of the image.

BL The 0^{th} row represents the bottom edge of the image and the 0^{th} column represents the left edge of the image.

TR The 0^{th} row represents the top edge of the image and the 0^{th} column represents the right edge of the image.

BR The 0^{th} row represents the bottom edge of the image and the 0^{th} column represents the right edge of the image.

RT The 0^{th} row represents the top right of the image and the 0^{th} column represents the top edge of the image.

RB The 0^{th} row represents the right edge of the image and the 0^{th} column represents the bottom edge of the image.

LT The 0^{th} row represents the left edge of the image and the 0^{th} column represents the top edge of the image.

LB The 0^{th} row represents the left edge of the image and the 0^{th} column represents the bottom edge of the image.

TIFF This is the value of the TIFF-compatible *Orientation* tag that would result in the same transformation. Windows Media Photo does not use the *Orientation* tag.

While an application may set a value for the Transformation tag, it must not depend on that value remaining the same. The Windows Media Photo codec may elect (typically during a re-encoding process) to apply the transformation specified by the tag to the image bits, resetting the tag to 0. In either case an application never has to perform the transformation specified by the tag. The Windows Media Photo decoder will use this tag to transform the image during decoding. An application only needs to set this flag to quickly apply a lossless transformation to an image.

If an application chooses to apply a transformation to a Windows Media Photo file by setting this metadata tag, it must take into account the current value of the tag. The table below specifies the correct tag setting. The first column specifies the current value of the tag. The header row specifies the additional transform that the application wants

to apply and the associated cell entry specifies the new value that must be stored in the tag, resulting in the combined transformation.

	Secondary Transform							
1st Transform	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	1	2	3	4	5	6	7
1	1	0	3	2	6	7	4	5
2	2	3	0	1	5	4	7	6
3	3	2	1	0	7	6	5	4
4	4	5	6	7	3	2	1	0
5	5	4	7	6	1	0	3	2
6	6	7	4	5	2	3	0	1
7	7	6	5	4	0	1	2	3

Whenever an image is encoded, it should be stored in an un-transformed orientation and the Transformation tag should be reset to zero.

If an application changes the *Transformation* metadata tag (effectively requesting a transform to be performed when the image is decoded) then the application must also make sure the values for ImageWidth and ImageHeight are correct (swap them if the transform includes a rotation.)

4.3.7 ImageDataDiscard

Tag = 48324 (0hBCC4)

Type = BYTE

Count = 1

If this optional metadata tag is present, it signifies the level of data that has been discarded from the image as a result of a compressed domain transcode to reduce the file size. Based on the value of this tag, the application should down-sample the image accordingly during decode.

This tag is set during a compressed domain transcode and should not be changed by application programs. It is provided as an external indication of the compressed data content of the image file that can be used to most effectively decode an image that has been reduced in file size using a compressed domain transcode operation.

Refer to the section below on compressed domain transcode operations for more information on image data discarding and the meaning of this metadata tag.

The allowable values are:

0 No image frequency data was discarded. This is a full resolution image. This is the default value and the assumed value if this optional tag is not present.

1 The FlexBits have been discarded, making an arbitrary reduction of the quality of the transcoded image without changing the effective resolution of the image. Down-sampling of the image during decode is not specifically recommended.

2 The HighPass frequency data band has been discarded (which also includes the FlexBits), effectively reducing the resolution of the transcoded image by a factor of 4 in both

dimensions. To render the best image quality, it is recommended that the image should be down-sampled by a factor of four in both dimensions when it is decoded.

3 Both the HighPass and LowPass frequency data bands have been discarded (which also includes the FlexBits), effectively reducing the resolution of the transcoded image by a factor of 16 in both dimensions. To render the best image quality, it is recommended that the image should be down-sampled by a factor of sixteen in both dimensions when it is decoded.

4.3.8 AlphaDataDiscard

Tag = 48325 (0hBCC5)

Type = BYTE

Count = 1

If this optional metadata tag is present, it signifies the level of data that has been discarded from the planar alpha channel as a result of a compressed domain transcode to reduce the file size. Based on the value of this tag, the application should down-sample the image during decode accordingly.

This tag is set during a compressed domain transcode and should not be changed by application programs. It is provided as an external indication of the compressed data content of the planar alpha channel contained in this image file and can be used to most effectively decode an image that has been reduced in file size using a compressed domain transcode operation. This tag is only valid if the image contains a planar alpha channel.

Refer to the section below on compressed domain transcode operations for more information on image data discarding and the meaning of this metadata tag.

The allowable values are:

0 No image frequency data was discarded. This is a full resolution image. This is the default value and the assumed value if this optional tag is not present.

1 The FlexBits have been discarded, making an arbitrary reduction of the quality of the transcoded image without changing the effective resolution of the image. Down-sampling of the image during decode is not specifically recommended.

2 The HighPass frequency data band has been discarded (which also includes the FlexBits), effectively reducing the resolution of the transcoded image by a factor of 4 in both dimensions. To render the best image quality, it is recommended that the image should be down-sampled by a factor of four in both dimensions when it is decoded.

3 Both the HighPass and LowPass frequency data bands have been discarded (which also includes the FlexBits), effectively reducing the resolution of the transcoded image by a factor of 16 in both dimensions. To render the best image quality, it is recommended that the image should be down-sampled by a factor of sixteen in both dimensions when it is decoded.

4.3.9 ImageType

Tag = 48132 (0hBC04)

Type = LONG

Count = 1

This optional metadata tag is used to specify the image type of each individual frame in a multi-frame file. This tag should not be used when a file contains a single frame.

The value of this tag is interpreted as a bit field, with a specific meaning associated with each of the 32 bits. At present, only two bits are defined; the remaining bits are reserved for future use and should be ignored or set to zero.

Bit 0 **Preview**: The frame is an alternate representation of another image in the file and is typically encoded as a reduced resolution thumbnail or preview. This frame, known as the Preview frame, may be used by applications that desire a simplified, high performance preview of the file.

A Windows Media Photo file should contain only one Preview frame. If multiple frames contain the ImageType metadata tag with this bit set, only the first frame should be used and any subsequent Preview frames should be ignored.

The Preview frame must be encoded in a basic pixel format. Ideally, it should be encoded in one of these two pixel formats:

- WICPixelFormat24bppBGR
- WICPixelFormat8bppGray

If any advanced pixel format is used, there is no guarantee that all decoders will be able to access the preview frame. To minimize the preview image size, an appropriate lossy compression is also recommended.

A Preview frame can be used in a number of different ways. It provides a high performance method to retrieve a displayable representation, especially for very large images. A preview can provide a basic pixel format representation of an image stored in an advanced pixel format. A preview frame can also be used to present an alternate view (such as a cropped region) of a larger image.

Applications should only create a Preview frame and set this metadata tag bit when it makes logical sense. Small images in basic pixel formats may not require a preview since it is just as easy to decode the entire image.

Bit 1 **Page**: The frame is an individual page in a sequence of pages within the file. Applications can make use of this ImageType value to identify which frames should be accessed when exposing multiple frames within a single file. If a file contains multiple frames, any frames that do not have an ImageType tag with this bit set should be considered application-specific private data and not exposed as a logical page or frame. Any operation that reads and re-saves such a file should preserve these frames in the same manner that any other unknown image or metadata content is preserved.

The TIFF 6.0 metadata tags PageNumber and PageName can optionally be included to describe the individual pages.

4.4 Descriptive Tags

4.4.1 ICCProfile

The ICCProfile contains information to correctly interpret the numeric color values contained in the image, in accordance with the format defined by the InterColor Consortium's InterColor Profile Format.

If this optional metadata tag is not present or has a Count of zero, RGB images (or their equivalent gray scale formats) are interpreted as serge for unsigned values or scRGB for signed or floating point values.

Tag = 34675 (0x8773)

Type = UNDEFINED

Count = the total number of bytes in the ICC Profile data record.

4.4.2 XMPMetadata

All the descriptive information for a Windows Media Photo file may be stored in XML syntax in accordance with the Adobe Extensible Metadata Platform (XMP) specification dated January 2004. (www.adobe.com/xmp) If XML-encoded XMP metadata is included, it is stored in this optional metadata tag.

Tag = 700 (0x2bc)

Type = BYTE

Count = the total number of bytes in the XML data record, including any terminator.

4.4.3 EXIFMetadata

All the descriptive information for a Windows Media Photo file may be stored in accordance with the EXIF 2.2 schema and syntax. This EXIF data is stored as an additional IFD table. The tag ID's and types in this table are as defined by EXIF 2.2. The value of this optional metadata tag specifies byte offset pointer to the beginning of the EXIF IFD table.

Tag = 34665 (0x8769)

Type = LONG

Count = 1

4.4.4 TIFF-compatible Descriptive Metadata Tags

Windows Media Photo also accepts the storage of optional descriptive metadata in TIFF compatible tags. Only tags that are purely descriptive in nature should be included. Any TIFF-compatible tags that describe the structure or content of the image information should not be used. The optional TIFF 6.0 compatible tags that are allowed include DocumentName, ImageDescription, Make, Model, Software, DateTime, Artist, HostComputer, PageName, PageNumber and Copyright. Refer to the TIFF 6.0 specification for details on these tags.

Chapter 5. Windows Image Codec (WIC) Application Program Interfaces

5.1 Overview

WMPhoto is an installable codec for Windows Media Photo files based on the Windows Imaging Component (WIC) installable codec architecture. It implements several of the WIC defined interfaces for codecs. The complete documentation for these interfaces can be found in the Windows Presentation Foundation imaging documentation.

5.1.1 Classes

WMPhoto defines the following C++ classes and implements the associated WIC interfaces as follows:

WMPhoto Class	*WIC Interface*
CWmpCodecInfo	IWICBitmapEncoderInfo
	IWICBitmapDecoderInfo
CWmpEncoder	IWICBitmapEncoder
CWmpDecoder	IWICBitmapDecoder
CWmpDecoderFrame	IWICBitmapFrameDecoder
	IWICBitmapSourceTransform
	IWICMetadataBlockReader
CWmpEncoderFrame	IWICBitmapFrameEncode
	IWICBitmapMetadataBlockWriter
	IPropertyBag2

Refer to the Windows Presentation Foundation imaging documentation for complete details on the WIC interfaces.

5.2 IPropertyBag2 Interface for Encoder Parameters

Parameters that control the image encoding process are specified using the Windows IPropertyBag2 interface. There are a set of canonical properties that apply to any image file codec type, and additional properties that are specific to WMPhoto. An application can provide basic control of the WMPhoto encoding process using the canonical properties or have more specific control using the codec-specific properties.

Using the IPropertyBag2 interface, an application can query the available encoder parameters. Each parameter also has a default value in the event it is not specified by the calling application. It is acceptable for an application to encode a file using default values by simply ignoring the encoder parameters and the associated IPropertyBag2 interface.

5.2.1 Canonical Encoder Parameter Properties

The Windows Image Component (WIC) interface expects all installable codecs to support a subset of these canonical encoder options:

Property Name	*VARTYPE*	*Value*
ImageQuality	VT_R4	0-1.0
CompressionQuality	VT_R4	0-1.0
Lossless	VT_BOOL	True/False
BitmapTransform	VT_UI1	WICBitmapTransformOptions

5.2.1.1 ImageQuality

0.0 means the lowest possible fidelity rendition and 1.0 means the highest fidelity, which for WMPhoto means lossless. This value maps to specific WMPhoto encoder parameters based on the following table:

ImageQuality	*Q (BD<=8)*	*Q (BD<=16)*	*Q (BD>16)*	*Subsample*	*Overlap*
>0.0 <0.4	105-IQ*100	185-IQ*180	245-IQ*240	4:4:4	2
>0.4 <1.0	105-IQ*100	185-IQ*180	245-IQ*240	4:4:4	1
1.0	1	1	1	4:4:4	0

The default value is 0.9.

5.2.1.2 CompressionQuality

0.0 means the least efficient compression scheme available, typically resulting in a fast encode but larger output. A value of 1.0 means the most efficient scheme available, typically taking more time to encode but producing smaller output.

WMPhoto does not support this canonical encoder option and it if it is present in the encoder options IPropertyBag2 parameter list, it will be ignored.

5.2.1.3 Lossless

Setting this parameter to TRUE enables mathematically lossless compression mode and overrides the ImageQuality parameter setting. The default value is FALSE.

5.2.1.4 BitMapTransform

This parameter is stored as the BitMapTransform value in the compressed bit stream header, controlling how the image will be transformed during decode. This property must be set to one or a combination of the allowable values of the WICBitmapTransformOptions enumerated type:

```
typedef enum WICBitmapTransformOptions {
    WICBitmapTransformRotate0 = 0x00000000,
    WICBitmapTransformRotate90 = 0x00000001,
    WICBitmapTransformRotate180 = 0x00000002,
    WICBitmapTransformRotate270 = 0x00000003,
    WICBitmapTransformFlipHorizontal = 0x00000008,
    WICBitmapTransformFlipVertical = 0x00000010,
    WICBITMAPTRANSFORMOPTIONS_FORCE_DWORD = CODEC_FORCE_DWORD
} WICBitmapTransformOptions;
```

The default value is 0 (WICBitmapTransformRotate0.)

5.2.2 WMPhoto-Specific Encoder Parameter Properties

WMPhoto supports the following encoder parameters via the IPropertyBag2 Interface:

Property Name	*VARTYPE*	*Value*
UseCodecOptions	VT_Bool	True/False (Default: False)
Quality	VT_UI1	1 – 255 (Default: 1)
Overlap	VT_UI1	0 – 2 (Default: 1)
Subsampling	VT_UI1	0 – 3 (Default: 3)
HorizontalTileSlices	VT_UI2	0 – 4095 (Default: 0)
VerticalTileSlices	VT_UI2	0 – 4095 (Default: 0)
FrequencyOrder	VT_Bool	True/False (Default: True)
InterleavedAlpha	VT_Bool	True/False (Default: False)
AlphaQuality	VT_UI1	1 – 255 (Default: 1)
CompressedDomainTranscode	VT_Bool	True/False (Default: True)
ImageDataDiscard	VT_UI1	0-3 (Default: 0)
AlphaDataDiscard	VT_UI1	0-4 (Default: 0)
IgnoreOverlap	VT_Bool	True/False (Default: False)

5.2.2.1 UseCodecOptions

If this parameter is TRUE, the Quality, Overlap and Subsampling parameters are used in place of the ImageQuality encoder canonical parameter. The default value is FALSE. When FALSE, the Quality, Overlap and Subsampling parameters are set based on a table lookup determined by the ImageQuality parameter.

The default value is FALSE.

5.2.2.2 Quality

This parameter controls the compression quality for the main image. A value of 1 sets lossless mode. Increasing values result in higher compression ratios and lower image quality.

The default value is 1.

5.2.2.3 Overlap

This parameter selects the optional overlap processing level:

0 No overlap processing is enabled.

1 One level of overlap processing is enabled, modifying 4x4 block encoded values based on values of neighboring blocks.

2 Two levels of overlap processing are enabled; in addition to the first level processing, encoded values of 16x16 macro blocks are modified based on the values of neighboring macro blocks.

The default value is 1.

5.2.2.4 Subsampling

This parameter only applies to RGB images. It enables additional compression in the chroma space, preserving luminance detail at the expense of color detail:

3 4:4:4 encoding preserves full chroma resolution.

2 4:2:2 encoding reduces chroma resolution to ½ of luminance resolution.

1 4:2:0 encoding reduces chroma resolution to ¼ of luminance resolution.

0 4:0:0 encoding discards all chroma content, preserving luminance only. Because the codec uses a slightly modified definition of luminance to improve performance, it is preferred to convert an RGB image to monochrome before encoding rather than use this chroma subsampling mode.

Any value greater than 3 returns an error. The default value is 3.

5.2.2.5 HorizontalTileSlices, VerticalTileSlices

These optional parameters specify the horizontal and vertical tiling of the image prior to compression encoding for the most optimal region decode performance. Dividing the image into rectangular tiles during encoding makes it possible to decode regions of the image without the need to process the entire compressed data stream. The default value of 0 specifies no subdivision, so the entire image is treated as a single tile. A value of 1 for each parameter will create a single horizontal and a single vertical division, effectively dividing the image into four equally sized tiles. The maximum value of 4095 for each parameter divides the image into 4096 tile rows with 4096 tiles per row. In other words, the parameter values equal the number of horizontal and vertical tiles (respectively) minus 1. A tile can never be smaller than 16 pixels in width or height, so the Windows Media Photo encoder may adjust this parameter to maintain the required minimum tile size. Because there is storage and processing overhead associated with each tile, these values should be chosen carefully to meet the specific scenario and unless there is a very specific reason, large numbers of small tiles should be avoided.

The default value for both parameters is 0.

5.2.2.6 Frequency Order

This parameter specifies that the image must be encoded in frequency order, with the lowest frequency data appearing first in the file, and image content grouped by its frequency rather than its spatial orientation. Organizing a file by frequency order provides the highest performance results for any frequency-based decoding, and is the preferred option. Device implementations of Windows Media Photo encoders may choose to organize a file in spatial order to reduce the memory footprint required during encoding.

The default value is TRUE and it is recommended that applications and devices always use frequency order unless there are performance or application-specific reasons to use spatial order.

5.2.2.7 InterleavedAlpha

Setting this parameter to true will instruct the codec to encode the alpha channel information as an additional interleaved channel, with no correlation to the image content channels. This mode is useful when it is important to decode alpha simultaneously with the image in a streaming scenario.

Setting this parameter to FALSE results in a planar alpha channel, encoded as a separate image with its own optional Quality value. A Planar alpha channel makes it possible to decode the

image data and the alpha channel independently. Interleaved alpha channels are only supported for certain RGB pixel formats. A Planar alpha channel can be associated with any image format that defines an alpha channel.

The default value is FALSE.

5.2.2.8 AlphaQuality

This parameter controls the compression quality for the planar alpha channel image. A value of 1 sets lossless mode. Increasing values result in higher compression ratios and lower image quality.

The default value is 1 (lossless compression.)

5.2.2.9 CompressedDomainTranscode

Windows Media Photo provides the unique ability to perform a number of useful file transformation operations without the need to actually decode the compressed data and re-encode it back to the destination file. Compressed domain operations are extremely efficient and avoid any additional quality loss typical when decoding and re-encoding a lossy-compressed image.

The following compressed domain operations are supported:

- Crop a region of the image
- Perform a rotation/flip transformation
- Discard frequency data (making it possible to create a smaller image file)
- Reorganize the image between spatial and frequency sequential order.

When using the WIC interfaces, a compressed domain transcode operation is performed by encoding an image with a Windows Media Photo decoder specified as the image source. Depending on the encoding options selected, the codec will use a compressed domain operation if possible. If an application chooses to explicitly inhibit any compressed domain transcode operations, the UseCodecOptions parameter should be set to TRUE and the CompressedDomainTranscode parameter should be set to FALSE.

When performing a compressed domain operation, only certain encoder parameter and property settings are allowed.

- The canonical encoder properties ImageQuality, CompressionQuality and Lossless are ignored.
- The WMPhoto-specific encoder properties Quality, Overlap, InterleavedAlpha and AlphaQuality are ignored.
- If present, the HorizontalSlices and VerticalSlices parameters must be set to zero. The tile size of an image cannot be changed as part of a compressed domain transcode.
- The image organization can be changed between frequency and spatial ordering by specifying the appropriate value of the FrequencyOrdering property. If this property is not present, no change is made to the image data order organization.
- A cardinal rotation and/or horizontal/vertical flip operation can be performed based on the value specified in the BitmapTransform canonical encoder parameter property.
- The image can be cropped by specifying the desired region using the WICRect parameter of the WriteSource encoder method.
- Image and/or alpha data can be discarded, shrinking the file size and effectively reducing the resolution of the transcoded image by specifying the appropriate values in the ImageDataDiscard and/or AlphaDataDiscard properties as described below.

5.2.2.10 ImageDataDiscard

This parameter is only valid if the CompressedDomainTranscode property is TRUE, otherwise it is ignored. ImageDataDiscard specifies the amount of image data to discard during a compressed domain transcode. If the image contains an interleaved alpha channel, this data discard will also apply to the alpha channel, with the exceptions as described in the section below.

The following values are allowed:

0 No image frequency data is discarded.

1 The FlexBits are discarded, making an arbitrary reduction of the quality of the transcoded image without changing the effective resolution of the image. The exact file size reduction or the specific quality reduction depends on numerous factors and cannot be specified or predicted. This value will return an error if specified for an interleaved alpha channel.

2 The HighPass frequency data band is discarded (which also includes the FlexBits), effectively reducing the resolution of the transcoded image by a factor of 4 in both dimensions. The actual dimensions of the transcoded image remain the same, but it will have lost all detail in each 4x4 block of pixels. Therefore, the transcoded image should be down-sampled accordingly whenever it is decoded.

3 Both the HighPass and LowPass frequency data bands are discarded (which also includes the FlexBits), effectively reducing the resolution of the transcoded image by a factor of 16 in both dimensions. The actual dimensions of the transcoded image remain the same, but it will have lost all detail in each 16x16 macroblock of pixels. Therefore, the transcoded image should be down-sampled accordingly whenever it is decoded.

5.2.2.11 AlphaDataDiscard

This parameter is only valid if the CompressedDomainTranscode property is TRUE and the image contains either a planar or interleaved alpha channel, otherwise it is ignored. It specifies the amount of alpha frequency data to discard during a compressed domain transcode. The following values are allowed for a planar alpha channel:

0 No image frequency data is discarded.

1 The FlexBits are discarded, making an arbitrary reduction of the quality of the planar alpha channel for the transcoded image without changing the effective resolution. The exact file size reduction or the specific quality reduction depends on numerous factors and cannot be specified or predicted.

2 The HighPass frequency data band is discarded (which also includes the FlexBits), effectively reducing the resolution of the transcoded image planar alpha channel by a factor of 4 in both dimensions. The actual dimensions of the transcoded image remain the same, but it will have lost all planar alpha channel detail in each 4x4 block of pixels. Therefore, the transcoded image should be down-sampled accordingly whenever it is decoded. Typically this value should only be set when the ImageDataDiscard property is set to the same value.

3 Both the HighPass and LowPass frequency data bands are discarded (which also includes the FlexBits), effectively reducing the resolution of the transcoded image by a factor of 16 in both dimensions. The actual dimensions of the transcoded image remain the same, but it will have lost all detail in each 16x16 macroblock of pixels. Therefore, the

transcoded image should be down-sampled accordingly whenever it is decoded. Typically this value should only be set when the ImageDataDiscard property is set to the same value.

4 The Alpha channel is completely discarded. The pixel format of the transcoded image is changed to reflect the removal of the alpha channel.

For images containing interleaved alpha channels, unless this property is set to 4, the alpha channel will be processed the same as the image data, according to the value of the ImageDataDiscard property. If this property is set to 4, the interleaved alpha channel will be completely discarded and the pixel format of the transcoded image will be changed accordingly.

5.2.2.12 IgnoreOverlap

This parameter is only valid if the CompressedDomainTranscode property is TRUE and a sub-region transcode of exactly one or more tiles is requested. The default operation for a region transcode (or decode) is to expand the requested region to include the surrounding pixels required for overlap decoding of the region edges. When this parameter is set to TRUE, the surrounding pixels are ignored and only the selected tile or tiles are extracted. Again, this requires that the requested region exactly match the coordinates of one or more tiles. If the source image is not tiled or if the requested region specifies any partial tiles, this parameter is ignored.

Support for the IgnoreOverap parameter is not currently implemented.

5.2.3 IPropertyBag2 Encoder/Decoder Options Usage

The component factory provides an easy creation point for creating an encoder options property bag. The property bag must be initialized during creation with all the encoder options. Option value ranges will be enforced on Write.

An application is given the encoder options bag during frame creation and must configure any values prior to initializing the encoder frame. For a UI driven app, it can offer a fixed UI for the canonical encoder options or an advanced, WMPhoto-specific view for the more detailed options. Changes can be made one at a time through the Write method, and any error will be reported through IErrorLog. The UI application should always re-read and display all options after making a change in case that change caused a cascading change. An application must be prepared to handle failed frame initialization for codecs that only provide minimal error reporting through their property bag.

```
virtual HRESULT STDMETHODCALLTYPE CreatePropertyBag(
     /* [in] */ PROPBAG2 *ppropOptions,
     /* [in] */ UINT cCount,
     /* [out] */ IPropertyBag2 **ppIPropertyBag2) PURE;
```

5.3 WMPhoto Implementation of IWICBitmapSourceTransform

The IWICSourceTransform interface provides and advanced option for decoding an image bit stream. Rather than just return a complete image using IWICBitMapFrameDecoder, this interface allows a variety of decoder options including:

- Decode a rectangular sub-region of the image
- Decode to a lower resolution
- Decode to a different pixel format
- Perform a transformation (rotation/flip) while decoding

While WIC enables these capabilities for all format types, WMPhoto provides codec level accelerated support for all these functions in many scenarios. Specifically:

5.3.1.1 DoesSupportTransform function

WMPhoto returns true for every transform option

5.3.1.2 GetClosestSize function

For requests that are less than ½ the dimension of the source image in both dimensions, WMPhoto returns the next largest integer image size that is evenly divisible by a factor of two. For all other requested sizes, WMPhoto returns the original image dimensions.

5.3.1.3 GetClosestPixelFormat function

WMPhoto returns the pixel format of the encoded image. In the future, the WMPhoto decoder may allow "no cost" transforms to other pixel formats during decode, but this functionality is not currently implemented.

5.3.1.4 CopyPixels function

WMPhoto accepts any requested region specified by the WICRect parameter and returns that portion of the image.

The uiWidth and uiHeight parameters must specify dimensions as returned by the GetClosestSize function. Any other values will return an error.

The pguidDstFormat parameter must specify the pixel format returned by the GetClosestPixelFormat function. Any other value will return an error.

WMPhoto accepts any allowable value for the dstTransform parameter. Note that the values allowed by WIC for this parameter are different than the values used by WMPhoto for the Transformation metadata tag.