Adobe Photoshop®
TIFF Technical Notes

March 6, 2002

This document describes the two compression schemes added by the Adobe Photoshop® "Advanced TIFF" options dialog when saving TIFF files. The two options are ZIP (Deflate/Inflate) and JPEG. The next two sections give the details of the TIFF file structure when saving with these options.

This document also describes the Image Source data tag written by Photoshop under tag number 37724.

Special considerations for Lab Images are discussed at the end of the document.

WARNING: TIFF readers that encounter SubIFD (tag number 330) and a bit depth other than 8 should ignore the SubIFD's found in the file.

Readers are advised to cross reference terms and discussions found in this document with the TIFF 6.0 specification (TIFF6.pdf), the TIFF Technical Note for Adobe PageMaker® 6.0 (TIFF-PM6.pdf), and the File Formats Specification for Adobe Photoshop® (Photoshop File Formats.pdf).

Deflate/Inflate Compression

Introduction

This section describes TIFF compression scheme 8, lossless Deflate compression using the zlib compressed data format:

Field: **Compression**
Tag: 259 (103.H)
Type: SHORT
Count: 1
Value: 8 Deflate compression, using zlib data format

Deflate is a byte-oriented compression scheme which can be applied to a TIFF image regardless of its **Photometric Interpretation**, **Planar Configuration**, or image data organization (strips or tiles). Each strip or tile (we will use the term *image segment* to refer to either) is compressed independently, and stored as a complete zlib data stream.

Note that the **Predictor** defined in (TIFF6.pdf Section 14) works well for both Deflate and LZW compression methods.

At this time, the Deflate compression methods are believed to be free of patent restrictions.

Reference

For a complete description of the zlib compressed data format, see the zlib specification (RFC1950, by L. Peter Deutsch and Jean-Loup Gailly, May 1996). For further details on the compression algorithm, see the Deflate specification (RFC1951, by L. Peter Deutsch and Jean-Loup Gailly, May 1996).

Implementation

In a TIFF image using this compression scheme, each image segment is compressed independently. Larger image segments should result in better compression ratios, up to a segment size of about 32K bytes (uncompressed size).

Each image segment consists of a single complete zlib data stream. The zlib data stream contains header and trailer information surrounding one or more compressed data blocks (see RFC1950). The compressed data blocks themselves are stored in the Deflate data format (see RFC1951).

The zlib header has the following structure:

Byte 0: Compression Method and Flags (CMF)
Bits 0-3 specify the compression method, and must be 8 (deflate).
Bits 4-7 specify the sliding window size; this may be any legal value.

Byte 1: Flag byte (FLG)
TIFF does not support preset dictionaries, so the FDICT flag must remain clear. All other flags may contain any legal value. There should not be a DICTID field in the zlib header.

The zlib trailer consists of a 4-byte CRC value, based on the *uncompressed* data. The algorithm for computing the CRC is defined in the zlib specification. As per the zlib specification, the CRC value is stored in big-endian byte order, regardless of the byte order specified in the TIFF file header.

The actual compressed data portion of the zlib data stream consists of one or more data blocks. The first block begins on a byte boundary, immediately following the header. Succeeding blocks are concatenated together, and do not necessarily begin on byte boundaries. The final block will be padded to the next byte boundary, prior to the CRC trailer. Each block begins with a 3-bit header. The first bit is a flag which indicates the final data block in the zlib data stream. The next two bits indicate the compression type for the block: uncompressed, LZ77 with fixed Huffman codes, or LZ77 with dynamic Huffman codes.

The compressed data may contain backward pointers to repeated byte strings. Although it should be obvious, we will state the following rule for completeness: while backward pointers may cross block boundaries (they may reference repeated byte strings up to 32K bytes away), they must not attempt to point outside of the current zlib data stream (i.e., they must not reference a block in another image segment). Further, these backward pointers bear no relationship whatsoever to TIFF offsets (which are often referred to as pointers in this document).

JPEG Compression

Introduction

This section describes TIFF compression scheme 7, a high-performance compression method for continuous-tone images. This TIFF compression method uses the international standard for image compression ISO/IEC 10918-1, usually known as "JPEG" (after the original name of the standards committee, Joint Photographic Experts Group). JPEG is a joint ISO/CCITT standard for compression of continuous-tone images.

The JPEG committee decided that because of the broad scope of the standard, no one algorithmic procedure was able to satisfy the requirements of all applications. Instead, the JPEG standard became a "toolkit" of multiple algorithms and optional capabilities. Individual applications may select a subset of the JPEG standard that meets their requirements.

The most important distinction among the JPEG processes is between lossy and lossless compression. Lossy compression methods provide high compression but allow only approximate reconstruction of the original image. JPEG's lossy processes allow the encoder to trade off compressed file size against reconstruction fidelity over a wide range. Typically, 10:1 or more compression of full-color data can be obtained while keeping the reconstructed image visually indistinguishable from the original. Much higher compression ratios are possible if a low-quality reconstructed image is acceptable. Lossless compression provides exact reconstruction of the source data, but the achievable compression ratio is much lower than for the lossy processes; JPEG's rather simple lossless process typically achieves around 2:1 compression of full-color data.

The most widely implemented JPEG subset is the "baseline" JPEG process. This provides lossy compression of 8-bit-per-channel data. Optional extensions include 12-bit-per-channel data, arithmetic entropy coding for better compression, and progressive/hierarchical representations. The lossless process is an independent algorithm that has little in common with the lossy processes.

It should be noted that the optional arithmetic-coding extension is subject to several US and Japanese patents. To avoid patent problems, use of arithmetic coding processes in TIFF files intended for inter-application interchange is discouraged.

All of the JPEG processes are useful only for "continuous tone" data, in which the difference between adjacent pixel values is usually small. Low-bit-depth source data is not appropriate for JPEG compression, nor are palette-color images good candidates. The JPEG processes work well on grayscale and full-color data.

Describing the JPEG compression algorithms in sufficient detail to permit implementation would require more space than we have here. Instead, we refer the reader to the References section.

What data is being compressed?

In lossy JPEG compression, it is customary to convert color source data to YCbCr and then down sample it before JPEG compression. This gives 2:1 data compression with hardly any visible image degradation, and it permits additional space savings within the JPEG compression step proper. However, these steps are not considered part of the ISO JPEG standard. The ISO standard is "color blind": it accepts data in any color space.

For TIFF purposes, the JPEG compression tag is considered to represent the ISO JPEG compression standard only. The ISO standard is applied to the same data that would be stored in the TIFF file if no compression were used. Therefore, if color conversion or down sampling are used, they must be reflected in the regular TIFF fields; these steps are not considered to be implicit in the JPEG compression `tag value.` **Photometric Interpretation** and related fields shall describe the color space actually stored in the file.

With the TIFF field definitions, down sampling is permissible for YCbCr and CIEL*a*b* data. Please see the **YCbCrSubSampling** field (TIFF6.pdf Section 21) and the **CIELabSubSampling** field (TIFF6.pdf Section 23) for more information. Note that the default values for these fields are not 1, 1; so *the default behavior is to apply down sampling*.

Implementers should note that many popular JPEG *codecs* (compressor/ decompressors) provide automatic color conversion and down sampling, so that the application may supply full-size RGB data which is nonetheless converted to down sampled YCbCr. This is an implementation convenience which does not excuse the TIFF control layer from its responsibility to know what is really going on. The **Photometric Interpretation** and sub sampling fields written to the file must describe what is actually in the file.

A JPEG-compressed TIFF file will typically have **Photometric Interpretation** = YCbCr and **YCbCrSubSampling** = [2,1] or [2,2], unless the source data was grayscale or CMYK.

Basic representation of JPEG-compressed images

JPEG compression works in either strip-based or tile-based TIFF files. Rather than repeating “strip or tile” constantly, we will use the term “segment” to mean either a strip or a tile.

When the **Compression** field has the value 7, each image segment contains a complete JPEG data stream which is valid according to the ISO JPEG standard (ISO/IEC 10918-1). Any sequential JPEG process can be used, including lossless JPEG, but progressive and hierarchical processes are not supported. Since JPEG is useful only for continuous-tone images, the **Photometric Interpretation** of the image shall not be 3 (palette color) nor 4 (transparency mask). The bit depth of the data is also restricted as specified below.

Each image segment in a JPEG-compressed TIFF file shall contain a valid JPEG data stream according to the ISO JPEG standard’s rules for interchange-format or abbreviated-image-format data. The data stream shall contain a single JPEG frame storing that segment of the image. The required JPEG markers within a segment are:

SOI	(must appear at very beginning of segment)
SOFn	
SOS	(one for each scan, if there is more than one scan)
EOI	(must appear at very end of segment)

The actual compressed data follows SOS; it may contain RSTn markers if DRI is used.

Additional JPEG “tables and miscellaneous” markers may appear between SOI and SOFn, between SOFn and SOS, and before each subsequent SOS if there is more than one scan. These markers include:

DQT	
DHT	
DAC	(not to appear unless arithmetic coding is used)
DRI	
APPn	(shall be ignored by TIFF readers)

COM (shall be ignored by TIFF readers)

DNL markers shall not be used in TIFF files. Readers should abort if any other marker type is found, especially the JPEG reserved markers; occurrence of such a marker is likely to indicate a JPEG extension.

The tables/miscellaneous markers may appear in any order. Readers are cautioned that although the SOFn marker refers to DQT tables, JPEG does not require those tables to precede the SOFn, only the SOS. Missing-table checks should be made when SOS is reached.

If no **JPEGTables** field is used, then each image segment shall be a complete JPEG interchange data stream. Each segment must define all the tables it references. To allow readers to decode segments in any order, no segment may rely on tables being carried over from a previous segment.

When a **JPEGTables** field is used, image segments may omit tables that have been specified in the **JPEGTables** field. Further details appear below.

The SOFn marker shall be of type SOF0 for strict baseline JPEG data, of type SOF1 for non-baseline lossy JPEG data, or of type SOF3 for lossless JPEG data. (SOF9 or SOF11 would be used for arithmetic coding.) All segments of a JPEGcompressed TIFF image shall use the same JPEG compression process, in particular the same SOFn type.

The data precision field of the SOFn marker shall agree with the TIFF **BitsPerSample** field. (Note that when **Planar Configuration**=1, this implies that all components must have the same **BitsPerSample** value; when **Planar Configuration**=2, different components could have different bit depths.) For SOF0 only precision 8 is permitted; for SOF1, precision 8 or 12 is permitted; for SOF3, precisions 2 to 16 are permitted.

The image dimensions given in the SOFn marker shall agree with the logical dimensions of that particular strip or tile. For strip images, the SOFn image width shall equal **ImageWidth** and the height shall equal **RowsPerStrip**, except in the last strip; its SOFn height shall equal the number of rows remaining in the **ImageLength**. (In other words, no padding data is counted in the SOFn dimensions.) For tile images, each SOFn shall have width **TileWidth** and height **TileHeight**; adding and removing any padding needed in the edge tiles is the concern of some higher level of the TIFF software. (The dimensional rules are slightly different when **Planar Configuration**=2, as described below.)

The ISO JPEG standard only permits images up to 65535 pixels in width or height, due to 2-byte fields in the SOFn markers. In TIFF, this limits the size of an individual JPEG-compressed strip or tile, but the total image size can be greater.

The number of components in the JPEG data stream shall equal **SamplesPerPixel** for P**lanar Configuration**=1, and shall be 1 for **Planar Configuration**=2. The components shall be stored in the same order as they are described at the TIFF field level. (This applies both to their order in the SOFn marker, and to the order in which they are scanned if multiple JPEG scans are used.) The component ID bytes are arbitrary so long as each component within an image segment is given a distinct ID. To avoid any possible confusion, we require that all segments of a TIFF image use the same ID code for a given component.

In **Planar Configuration** 1, the sampling factors given in SOFn markers shall agree with the sampling factors defined by the related TIFF fields (or with the default values that are specified in the absence of those fields).

When DCT-based JPEG is used in a strip TIFF file, **RowsPerStrip** is required to be a multiple of 8 times the largest vertical sampling factor, i.e., a multiple of the height of an interleaved MCU. (For simplicity of specification, we require this even if the data is not actually interleaved.) For example, if **YCbCrSubSampling** = [2,2] then **RowsPerStrip** must be a multiple of 16. An exception to this rule is made for single-strip images (**RowsPerStrip** >= **ImageLength**): the exact value of **RowsPerStrip** is unimportant in that case. This rule ensures that no data padding is needed at the bottom of a strip, except

perhaps the last strip. Any padding required at the right edge of the image, or at the bottom of the last strip, is expected to occur internally to the JPEG codec.

When DCT-based JPEG is used in a tiled TIFF file, **TileLength** is required to be a multiple of 8 times the largest vertical sampling factor, i.e., a multiple of the height of an interleaved MCU; and **TileWidth** is required to be a multiple of 8 times the largest horizontal sampling factor, i.e., a multiple of the width of an interleaved MCU. (For simplicity of specification, we require this even if the data is not actually interleaved.) All edge padding required will therefore occur in the course of normal TIFF tile padding; it is not special to JPEG.

Lossless JPEG does not impose these constraints on strip and tile sizes, since it is not DCT-based.

Note that within JPEG data streams, multibyte values appear in the MSB-first order specified by the JPEG standard, regardless of the byte ordering of the surrounding TIFF file.

JPEGTables field

The only auxiliary TIFF field added for Compression=7 is the optional **JPEGTables** field. The purpose of **JPEGTables** is to predefine JPEG quantization and/or Huffman tables for subsequent use by JPEG image segments. When this is done, these rather bulky tables need not be duplicated in each segment, thus saving space and processing time. **JPEGTables** may be used even in a single segment file, although there is no space savings in that case.

Field: **JPEGTables**
Status: optional
Tag: 347 (15B.H)
Type: UNDEFINED
Count: number of bytes in tables data stream, typically a few hundred
Value: JPEGTables provides default JPEG quantization and/or Huffman tables which are used whenever a segment data stream does not contain its own tables, as specified below.
Default: none

Notice that the **JPEGTables** field is required to have type code UNDEFINED, not type code BYTE. This is to cue readers that expanding individual bytes to short or long integers is not appropriate. A TIFF reader will generally need to store the field value as an uninterpreted byte sequence until it is fed to the JPEG decoder.

Multibyte quantities within the tables follow the ISO JPEG convention of MSBfirst storage, regardless of the byte ordering of the surrounding TIFF file.

When the **JPEGTables** field is present, it shall contain a valid JPEG "abbreviated table specification" data stream. This data stream shall begin with SOI and end with EOI. It may contain zero or more JPEG "tables and miscellaneous" markers, namely:

DQT	
DHT	
DAC	(not to appear unless arithmetic coding is used)
DRI	
APPn	(shall be ignored by TIFF readers)
COM	(shall be ignored by TIFF readers)

Since JPEG defines the SOI marker to reset the DAC and DRI state, these two markers' values cannot be carried over into any image data stream, and thus they are effectively no-ops in the **JPEGTables** field. To

avoid confusion, it is recommended that writers not place DAC or DRI markers in **JPEGTables**. However, readers must properly skip over them if they appear.

When **JPEGTables** is present, readers shall load the table specifications contained in **JPEGTables** before processing image segment data streams. Image segments may simply refer to these preloaded tables without defining them.

An image segment can still define and use its own tables, subject to the restrictions below. An image segment may not redefine any table defined in **JPEGTables**. (This restriction is imposed to allow readers to process image segments in random order without having to reload **JPEGTables** between segments.) Therefore, use of **JPEGTables** divides the available table slots into two groups: "global" slots are defined in **JPEGTables** and may be used but not redefined by segments; "local" slots are available for local definition and use in each segment. To permit random access, a segment may not reference any local tables that it does not itself define.

Special considerations for Planar Configuration 2

In **Planar Configuration** 2, each image segment contains data for only one color component. To avoid confusing the JPEG codec, we wish the segments to look like valid single-channel (i.e., grayscale) JPEG data streams. This means that different rules must be used for the SOFn parameters.

In **Planar Configuration** 2, the dimensions given in the SOFn of a sub sampled component shall be scaled down by the sampling factors compared to the SOFn dimensions that would be used in p**lanar Configuration** 1. This is necessary to match the actual number of samples stored in that segment, so that the JPEG codec doesn't complain about too much or too little data. In strip TIFF files the computed dimensions may need to be rounded up to the next integer; in tiled files, the restrictions on tile size make this case impossible.

Furthermore, all SOFn sampling factors shall be given as 1. (This is merely to avoid confusion, since the sampling factors in a single-channel JPEG data stream have no real effect.)

Any down sampling will need to happen externally to the JPEG codec, since JPEG sampling factors are defined with reference to the full-precision component. In **Planar Configuration** 2, the JPEG codec will be working on only one component at a time and thus will have no reference component to down sample against.

Minimum requirements for TIFF/JPEG

ISO JPEG is a large and complex standard; most implementations support only a subset of it. Here we define a "core" subset of TIFF/JPEG which readers must support to claim TIFF/JPEG compatibility. For maximum cross-application compatibility, we recommend that writers confine themselves to this subset unless there is very good reason to do otherwise.

Use the ISO baseline JPEG process: 8-bit data precision, Huffman coding, with no more than 2 DC and 2 AC Huffman tables. Note that this implies **BitsPerSample**
= 8 for each component. We recommend deviating from baseline JPEG only if 12-bit data precision or lossless coding is required.

Use no sub sampling (all JPEG sampling factors = 1) for color spaces other than YCbCr and CIEL*a*b*. For YCbCr and CIEL*a*b*, use one of the following choices:

ChromaSubSampling field	JPEG sampling factors
1,1	1h1v, 1h1v, 1h1v
2,1	2h1v, 1h1v, 1h1v
2,2 (default value)	2h2v, 1h1v, 1h1v

We recommend that RGB source data be converted to YCbCr for best compression results. Other source data color spaces should probably be left alone. Minimal readers need not support JPEG images with color spaces other than YCbCr and grayscale (**Photometric Interpretation** = 6 or 1).

A minimal reader also need not support JPEG YCbCr images with no default values of **YCbCrCoefficients** or **ChromaPositioning**, nor with values of **ReferenceBlackWhite** other than [0,255,128,255,128,255]. (These values correspond to the RGB<=>YCbCr conversion specified by JFIF, which is widely implemented in JPEG codecs.)

If any sub sampling is used, **Planar Configuration**=1 is preferred to avoid the possibly-confusing requirements of **Planar Configuration**=2. In any case, readers are not required to support **Planar Configuration**=2.

If possible, use a single interleaved scan in each image segment. This is not legal JPEG if there are more than 4 **SamplesPerPixel** or if the sampling factors are such that more than 10 blocks would be needed per MCU; in that case, use a separate scan for each component. (The recommended color spaces and sampling factors will not run into that restriction, so a minimal reader need not support more than one scan per segment.)

To claim TIFF/JPEG compatibility, readers shall support multiple-strip TIFF files and the optional **JPEGTables** field; it is not acceptable to read only single data stream files. Support for tiled TIFF files is strongly recommended but not required.

Photoshop 7.0 writes **ChromaSubSampling** of [1,1] for quality settings above 6 with **ChromaPositioningValue** = 2.

Other recommendations for implementers

The TIFF tag **Compression**=7 guarantees only that the compressed data is represented as ISO JPEG data streams. Since JPEG is a large and evolving standard, readers should apply careful error checking to the JPEG markers to ensure that the compression process is within their capabilities. In particular, to avoid being confused by future extensions to the JPEG standard, it is important to abort if unknown marker codes are seen.

The point of requiring that all image segments use the same JPEG process is to ensure that a reader need check only one segment to determine whether it can handle the image. For example, consider a TIFF reader that has access to fast but restricted JPEG hardware, as well as a slower, more general software implementation. It is desirable to check only one image segment to find out whether the fast hardware can be used. Thus, writers should try to ensure that all segments of an image look as much “alike” as possible: there should be no variation in scan layout, use of options such as DRI, etc. Ideally, segments will be processed identically except perhaps for using different local quantization or entropy-coding tables.

Writers should avoid including “noise” JPEG markers (COM and APPn markers). Standard TIFF fields provide a better way to transport any non-image data. Some JPEG codecs may change behavior if they see an APPn marker they think they understand; since the TIFF spec requires these markers to be ignored, this behavior is undesirable.

It is possible to convert an interchange-JPEG file (e.g., a JFIF file) to TIFF simply by dropping the interchange data stream into a single strip. (However, designers are reminded that the TIFF spec discourages huge strips; splitting the image is somewhat more work but may give better results.) Conversion from TIFF to interchange JPEG is more complex. A strip-based TIFF/JPEG file can be converted fairly easily if all strips use identical JPEG tables and no RSTn markers: just delete the overhead markers and insert RSTn markers between strips. Converting tiled images is harder, since the data will usually not be in the right order (unless the tiles are only one MCU high). This can still be done losslessly, but it will require undoing and redoing the entropy coding so that the DC coefficient differences can be updated.

There is no default value for **JPEGTables**: standard TIFF files must define all tables that they reference. For some closed systems in which many files will have identical tables, it might make sense to define a default **JPEGTables** value to avoid actually storing the tables. Or even better, register a private field for selecting one of N default **JPEGTables** settings, so as to allow for future expansion. *Either of these suggestions must be regarded as a private extension that will render the files unreadable by other applications.*

References

[1] Wallace, Gregory K. “The JPEG Still Picture Compression Standard”, Communications of the ACM, April 1991 (vol. 34 no. 4), pp. 30-44.

This is the best short technical introduction to the JPEG algorithms. It is a good overview but does not provide sufficiently detailed information to write an implementation.

[2] Pennebaker, William B. and Mitchell, Joan L. “JPEG Still Image Data Compression Standard”, Van Nostrand Reinhold, 1993, ISBN 0-442-01272-1. 638pp.

This textbook is by far the most complete exposition of JPEG in existence. It includes the full text of the ISO JPEG standards (DIS 10918-1 and draft DIS 10918-2). No would-be JPEG implementer should be without it.

[3] ISO/IEC IS 10918-1, “Digital Compression and Coding of Continuous-tone Still Images, Part 1: Requirements and guidelines”, February 1994. ISO/IEC DIS 10918-2, “Digital Compression and Coding of Continuous-tone Still Images, Part 2:Compliance testing”, final approval expected 1994.

These are the official standards documents. Note that the Pennebaker and Mitchell textbook are likely to be cheaper and more useful than the official standards.

Image Source Data

Introduction

This section describes the Adobe Photoshop specific image source data tag. This section has the layer and mask information found in a typical layered Photoshop file.

Field: **ImageSourceData**
Tag: 37724 (935C.H)
Type: UNDEFINED
Count: number of bytes for section

The section starts with a character string of "Adobe Photoshop Document Data Block" including the null termination character.

The remainder of the count, after subtracting the length of the above mentioned string, is a series of tagged data types in the following format:

4 bytes	Signature	'8BIM'
4 bytes	Type	various types (see below)
4 bytes	Length	length in bytes, variable for each type, padded to a 4 byte offset

The various types are mentioned here with further documentation in the Photoshop File Formats.pdf.
The available types are:

'Layr'	Layer Data	
'LMsk'	User Mask	Same as Global layer mask info table
'Patt'	Pattern	
'Anno'	Annotations	

L*a*b* Images

Introduction

CIE L*a*b* is a color space that is colorimetric, has separate lightness and chroma channels, and is approximately perceptually uniform. It has excellent applicability for device-independent manipulation of continuous tone images. These attributes make it an excellent choice for many image editing and archiving functions. 1976 CIEL*a*b* is represented as a Euclidean space with the following three quantities plotted along axes at right angles: L^* representing lightness, a^* representing the red/green axis, and b^* representing the yellow/blue axis. The formulas for 1976 CIE $L^*a^*b^*$ follow:

$L^* = 116(Y/Y_n)^{1/3}\text{-}16$ for $Y/Y_n > 0.008856$
$L^* = 903.3(Y/Y_n)$ for $Y/Y_n <= 0.008856$ *see note below.
$a^* = 500[(X/X_n)^{1/3}\text{-}(Y/Y_n)^{1/3}]$
$b^* = 200[(Y/Y_n)^{1/3}\text{-}(Z/Z_n)^{1/3}]$.

where X_n,Y_n, and Z_n are the CIE X, Y, and Z tristimulus values of an *appropriate* reference white. Also, if any of the ratios X/X_n, Y/Y_n, or Z/Z_n is equal to or less than 0.008856, it is replaced in the formulas with

$7.787F + 16/116$,

where F is X/X_n, Y/Y_n, or Z/Z_n, as appropriate (note: these low-light conditions are of no relevance for most document-imaging applications).

Scaling the 0-100 range of L^* into 256 levels provides lightness steps that are less than half the size of a "just noticeable difference" in the 8 bit encoding. Limiting the theoretically unbounded a^* and b^* ranges to +/- 127 allows encoding in 8 bits without eliminating any but the most saturated self-luminous colors

The TIFF CIELAB Fields

Field: **PhotometricInterpretation**
Status: required
Tag: 262 (106.H)
Type: SHORT
Count: 1
Value: This Field indicates the color space of the image. The new values are:
8 1976 CIE $L^*a^*b^*$, normal encoding
9 ICCLab encoding

For CIELab (PhotometricInterpretation = 8), the L* component is encoded in 8 bits as an unsigned integer range [0,255], and encoded in 16 bits as an unsigned integer range [0,65535]. The a* and b* components are encoded in 8 bits as signed integers range [-128,127], and encoded in 16 bits as signed integers range [-32768,32767]. The 8 bit chrominance values are exactly equal to the 1976 CIE a* and b* values, while the 16 bit values are equal to 256 times the 1976 CIE a* and b* values.

For ICCLab (PhotometricInterpretation = 9), the L* component is encoded in 8 bits as an unsigned integer range [0,255], and encoded in 16 bits as an unsigned integer range [0,65280]. The a* and b* components are encoded in 8 bits as unsigned integers range [0,255], and encoded in 16 bits as unsigned integers range [0,65535]. The 8 bit chrominance values are exactly equal to the 1976 CIE a* and b* values plus 128, while the 16 bit values are equal to 256 times the 1976 CIE a* and b* values plus 32768 (this is also 256 times the 8 bit encoding). PhotometricInterpretation 9 is designed to match the encoding used by the ICC profile specification.

Usage of other Fields

BitsPerSample: 8 or 16
SamplesPerPixel - ExtraSamples: 3 for *L*a*b**, 1 implies *L** only, for monochrome data.
Compression: same as other multi-bit formats.
PlanarConfiguration: both chunky and planar data could be supported.
WhitePoint: does not apply
PrimaryChromaticities: does not apply.
TransferFunction: does not apply
Alpha Channel information will follow the lead of other data types.

Important note:

Some application vendors developed (mutually incompatible) proprietary encodings for 16 bit LAB data in TIFF images before the release of this update. Unfortunately, there is no simple way to determine whether a file uses the standard encoding or a proprietary encoding other than opening the image and verifying its' appearance in an application known to use the standard encoding specified above. Legacy files which used these incompatible encodings will have to be converted to the standard encoding using updated versions of the software that created them or conversion utilities supplied by the company that created them.