Adobe PageMaker® 6.0 TIFF Technical Notes

September 14, 1995

A PDF version of this specification can be found at:

http://www.adobe.com/Support/TechNotes.html
or
ftp://ftp.adobe.com/pub/adobe/DeveloperSupport/TechNotes/PDFfiles

Free Acrobat Readers (PDF viewing/printing) can be downloaded from:

http://www.adobe.com/Software/Acrobat
or
ftp://ftp.adobe.com/pub/adobe/Applications/Acrobat

Contact:

devsup-person@adobe.com.

Contents

TIFF Tech Note 1: TIFF Trees 4
TIFF Tech Note 2: Clipping Path 6
TIFF Tech Note 3: Indexed Images 11
*TIFF Tech Note 4: ICC L*a*b* 13*
Other Major PageMaker 6.0 TIFF changes 14
PageMaker 6.0 TIFF Support Worksheet 16

TIFF Tech Note 1: TIFF Trees

Motivation

TIFF has always supported what amounts to a singly linked list of IFD's in a single TIFF file, via the "next IFD pointer," though most applications currently ignore any IFD beyond the first one.

Probably the best use for a linked list of IFD's is when you want to store multiple different but related images in the same file—a 'burst' of images from a camera, for example.

But suppose we want to define low-res "thumbnails" for each of the images in a burst of images. Where should we put those?

Solution

If we had the concept of a tree within a TIFF file, we would have a natural way to associate a main or "parent" image with a subordinate or "child" image such as a thumbnail.

One way to create a tree structure within a TIFF file is to define a new tag, which we will call **SubIFDs**. The value of the tag points to one or more "child" IFD structures.

Use the NextIFD pointer if your application requires that multiple non-identical images be stored in the same TIFF file—a burst of images from a camera, or a multi-page fax transmission, for example. Use the **SubIFDs** tag for pointing to ancillary images: images that modify or add information to or otherwise "help" the Parent image. The typical examples are thumbnails and other subsampled versions.

New Tag

SubIFDs

Tag = 330 (14A)

Type = LONG or "IFD" (Type = 13). "IFD" is preferred.

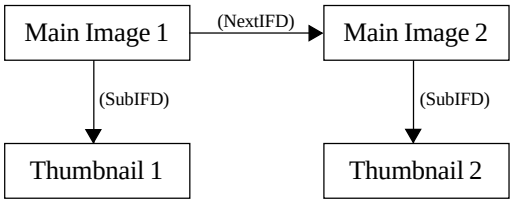
N = number of child IFDs

Each value is an offset (from the beginning of the TIFF file, as always) to a child IFD. Child images provide extra information for the parent image—such as a subsampled version of the parent image.

TIFF data type 13, "IFD," is otherwise identical to LONG, but is only used to point to other valid IFDs.

Examples

Let's revisit our example. We have a burst of images, plus thumbnails. The IFDs in our file would look like this:



The **SubIFDs** tag is used in both Main Image 1 and Main Image 2. In each Main image, the **SubIFDs** tag has one value, which points to the beginning of the IFD structure for its Thumbnail child image.

The NextIFD value of Main Image 1 points to Main Image 2. The NextIFD value of Main Image 2 is null.

If there is more than 1 child image for a given parent image, the NextIFD value of Child #1 must point to Child #2, and so on. The last Child's NextIFD value must be zero.

This completes the TIFF Trees technical note.

TIFF Tech Note 2: Clipping Path

Motivation

There is currently no standard way of including a PostScript-style clipping path in a TIFF file. The concept of image mask, which is outlined in the TIFF 6.0 specification, can in theory accomplish the same goal, but there are a number of advantages to using clipping paths.

One advantage is that a geometric clipping path can be readily implemented on PostScript printers. In contrast, a raster-based image mask has no PostScript equivalent. (The PostScript “imagemask” operator can only be used for making marks in a single color.)

Another advantage is that a geometric clipping path gives sharp, clean edges. A raster-based image mask has to be of a very high resolution (approximately the resolution of the printer or imagesetter) in order to accomplish the same thing, and therefore would typically take up much more space in a file, even with compression.

Implementation

ClipPath

Tag = 343 (157.H)

Type = BYTE

N = total number of BYTEs in the ClipPath

A TIFF ClipPath is intended to mirror the essentials of PostScript’s path creation functionality, so that the operators listed below can be easily translated into PostScript, and, conversely, any PostScript path can be represented as a TIFF ClipPath. However, the TIFF ClipPath list of operators is not identical to the current list of PostScript operators; for simplicity, some of the PostScript variants have been dropped, and a few operators, such as polyto and rpolyto have been added. polyto and rpolyto were added in order to make complex TIFF polygonal clip paths more compact.

A TIFF ClipPath is made up of a header, followed by a sequence of operators and operands.

Header

The byte order of multi-byte operands is determined by the first 16-bit word of the ClipPath field, which must be ‘II’ or ‘MM’. The 2nd word of ClipPath is the ClipPath Version number, which is zero. Words 3 through 8 are reserved, and must be zero.

Commands

Following the header is a sequence of operators and operands. An operator with its operands is called a "command." A command is made up of: OperatorID, DataType, Length, and the operands.

OperatorID is an 8-bit unsigned value (i.e., a TIFF 'BYTE').

DataType is also a BYTE, and must be one of the standard TIFF data types. Only SBYTE = DataType 6, SSHORT = 8, and SLONG = 9 should be used at this time.

Next comes an 8-bit reserved byte, which must be zero.

Length comes next, a BYTE which gives the number of bytes (not number of SBYTEs, SSHORTs, or SLONGs) used by the operands. Note that this is different from the rule used in TIFF IFDs. For example, if the operands for a command consist of 2 SLONGs, Length is 8, not 2.

The Length field is followed by the operands, which are usually absolute or relative coordinates.

The next command begins immediately after the last operand of the previous command.

Commands need not begin on even-byte boundaries.

The operators are taken from the following list:

OperatorID	*operands*	*name of operator*
0		ignored. no operation is performed.
1		**newpath**
2		**closepath**
3	x y	**moveto**
4	x y	**lineto**
5		(reserved)
6		(reserved)
7		(reserved)
8	x1 y1 x2 y2 x3 y3	**curveto**
9	x1 y1 x2 y2…	**polyto**
10	dx1 dy1 dx2 dy2…	**rpolyto**
11-255		(reserved)

The operators, in general

Unless otherwise noted, TIFF ClipPath operators have the same path construction effect as their PostScript counterparts. See the Adobe "Red Book" (PostScript Language Reference Manual) for details.

newpath, closepath

The first operator in a TIFF ClipPath must be **newpath**; the last must be **closepath**. **newpath** and **closepath** have no operands, so their Length will be zero, and their DataType is ignored.

Subpaths can be created by using **closepath**, followed by **moveto** and other path creation commands.

newpath erases all subpaths of the current path, so the only call to **newpath** should be the first command. Additional calls to **newpath** within a TIFF ClipPath are illegal.

moveto, lineto

moveto and **lineto** each have two operands.

Coordinates for **moveto** and **lineto** are absolute. **lineto** draws starting from the current point.

The first command after **newpath** must be **moveto**, since the other commands depend on having a valid currentpoint.

curveto

curveto has six operands. Like the other commands listed above, **curveto** is defined as it is in the PostScript Language Reference Manual, and describes a cubic Bezier arc.

polyto, rpolyto

Coordinates for **polyto** are absolute. Coordinates for **rpolyto** are relative to the previous coordinate pair. The current point is the first point of a polyline path segment created by **polyto** or **rpolyto**. **polyto** and **rpolyto** set the current point to the last point of the polyline.

Example: draw a square path whose upper left corner is (100,100), and lower right corner is (300, 300), using polyto:

```
moveto  100 100
polyto  300 100   300 300   100 300
closepath
```

Example: same, using **rpolyto**:

```
moveto   100 100
rpolyto  200 0   0 200   -200 0
closepath
```

polyto is not a PostScript operator, but is readily translated into multiple **lineto** PostScript commands. **rpolyto** is not a PostScript operator, but is readily translated into multiple **rlineto** PostScript commands.

coordinate system

The origin of the coordinate system for ClipPath is the upper left corner of the image. The coordinates are signed integers (SBYTE, SSHORT, or SLONG). X coordinates increase left-to-right, and Y coordinates increase top-to-bottom. (Note that this is not the same as default PostScript user space.)

The dimension of a coordinate represents the numerator of the fraction: numerator / XClipPathUnits, where a fraction of one (1.0) represents the width and height of the image. See the XClipPathUnits field description, below. So, for example, if

XClipPathUnits = 1000, a clipping path that surrounds the entire image is given by:

```
% start the path
newpath
% initial point is the upper left corner
moveto  0  0
% then go to the upper right, then lower right, then lower left
polyto   1000 0   1000 1000    0 1000
% close the path, connecting the lower left and upper left corners
closepath
```

Some or all ClipPath coordinates may lie outside the bounds of the image. This may be useful when clipping to certain shapes. This factor should be taken into account when choosing an appropriate value for XClipPathUnits, otherwise the units could overflow.

This coordinate system was chosen because it is quite compact, does not require floating point data storage and conversion, and the ClipPath does not have to be changed if the image is upsampled or downsampled—even if it is resampled differently in each axis.

Insidedness

For complex clippaths (e.g., a path that intersects itself or has one subpath that encloses another), the interpretation of "inside" is determined by the *even-odd rule.* This rule determines the 'insidedness' of a point by drawing a ray from that point in any direction and counting the number of path segments that the ray crosses. If this number is odd, the point is inside; if even, the point is outside.

The even-odd fill rule was chosen for compatibility with Apple QuickDraw, Adobe Photoshop and Microsoft Windows.

XClipPathUnits

Tag = 344 (158.H)

Type = DWORD

N = 1

The number of units that span the width of the image, in terms of integer ClipPath coordinates.

All horizontal ClipPath coordinates will be divided by this value in order to get a number that is (usually) between 0.0 and 1.0, where 0.0 represents the left side of the image and 1.0 represents the right side of the image.

Note that the choice of value for XClipPathUnits will influence the choice of DataType for the commands, since SSHORT or even SBYTE values may be usable if XClipPathUnits is smaller, while SLONG will be required if XClipPathUnits is larger.

Required for every TIFF ClipPath. No default.

YClipPathUnits

Tag = 345 (159.H)

Type = DWORD

N = 1

The number of units that span the height of the image, in terms of integer ClipPath coordinates.

All vertical ClipPath coordinates will be divided by this value in order to get a number that is (usually) between 0.0 and 1.0, where 0.0 represents the top of the image and 1.0 represents the bottom of the image.

Use this if you want to be able to specify your ClipPath coordinates using integer values that match the aspect ratio of an image.

Optional. Default is YClipPathUnits = XClipPathUnits.

This completes the TIFF Clipping Path technical note.

TIFF Tech Note 3: Indexed Images

Motivation

In the TIFF 6.0 specification, support for indexed images is currently limited to the RGB color space, and is signalled by PhotometricInterpretation = PaletteColor. The purpose of this addendum is to broaden the support for indexed images to include support for any color space, so that indexed images can be converted between color spaces by merely changing their color maps, without having to convert to chunky image data.

New Fields

Indexed

Tag = 346 (15A.H)

Type = SHORT

N = 1

1 = Indexed.

0 = not.

Indexed images are images where the “pixels” do not represent color values, but rather an index (usually 8-bit) into a separate color table, the ColorMap. ColorMap is required for an Indexed image.

The PhotometricInterpretation type of PaletteColor may still be used, and is equivalent to specifying an RGB image with the Indexed flag set, a suitable ColorMap, and SamplesPerPixel = 1.

Do not use both the Indexed flag and PhotometricInterpretation = PaletteColor for the same image.

Default is 0 (not indexed).

Changes to Existing Fields

ColorMap

ColorMap can now contain the color map for any color space.

The number of components in the ColorMap depends on the color space: for RGB and CIELab images there are 3 components, for CMYK there are 4 components,

and for other separated images there are NumberOfInks components. (See Section 16 of the TIFF 6.0 specification).

This completes the TIFF Indexed Images technical note.

TIFF Tech Note 4: ICC L*a*b*

Motivation and Implementation

Several recent implementations of CIE L*a*b*, including the InterColor Profile Format, encode the normal range of [-128, 127] for the a* and b* components into the range [0,255]. The new TIFF PhotometricInterpretation value of ICCLab = 9 is used for such data.

To go from the current (see Section 23 of the TIFF 6.0 specification) TIFF CIE L*a*b* format of -128 to 127 to the ICC format of 0 to 255, add 128 to the a* and b* values.

L* is encoded identically in both formats.

See the InterColor Consortium's InterColor Profile Format document.

This completes the TIFF ICC L*a*b* technical note.

Other Major PageMaker 6.0 TIFF changes

Adobe Photoshop Clipping Paths

In addition to supporting the TIFF Clipping Paths specified by Technical Note 2, PageMaker 6.0 also supports closed clipping paths specified in Photoshop 3 and stored in Photoshop's private TIFF tag. See the Adobe Photoshop Developer Kit for further information.

ANSI IT8 TIFF/IT

PM6 supports ANSI IT8 TIFF/IT CMYK and Linework files.

Dot Range

PM6 supports the DotRange tag. See Section 16 of the TIFF 6.0 specification.

Faster Planar CMYK Separations

Previous versions of PageMaker supported planar images (PlanarConfiguration = 2), but PM6 is optimized for printing planar CMYK and hifi images. You will notice one-fourth of the disk activity and resulting faster print times if you make your CMYK/hifi images planar.

Multi-Ink and HiFi

PM6 supports hifi and other multi-ink images. See Section 16 of the TIFF 6.0 specification.

Low-resolution preview images

PageMaker 6.0 uses the SubIFDs tag for storing an RGB or CMYK low-resolution preview image, when creating pre-separated hifi and CMYK images. PM6 uses this preview image for subsequent on-screen display.

To create a pre-separated image in PM6, choose the "Element:Image:Save for separation…" menu item.

See the TIFF Trees section for further information on the SubIFDs tag.

OPI-Related Tags

PM6 supports two OPI-related tags. For further information, see the Adobe OPI 2.0 specification.

ImageID

Tag = 32781 (800D)

Type = ASCII

ImageID is the full pathname of the original, high-resolution image, or any other identifying string that uniquely identifies the original image.

The high-resolution image is not required to be in TIFF format. It can be in any format that an OPI Consumer wishes to support.

OPIProxy

Tag = 351 (15F)

Type = SHORT

N = 1

OPIProxy gives information concerning whether this image is a low-resolution proxy of a high-resolution image.

A value of 1 means that a higher-resolution version of this image exists, and the name of that image is found in the ImageID tag (see above).

If this tag does not exist, or the value is 0, then a higher-resolution version of this image does not exist.

PageMaker 6.0 TIFF Support Worksheet

Product:	Adobe® PageMaker®
Version:	6.00
Date:	14_Sept_95
Contact name:	Adobe Developers Association Hotline
Contact phone:	415-961-4111
Contact fax #:	415-967-9231
Contact email:	devsup-person@adobe.com

Tag	#	Import Behavior
[New]SubfileType	254	Ignored on import, but written to new TIFF files.
ImageWidth	256	no restriction. can be LONG.
ImageLength	257	no restriction. can be LONG.
BitsPerSample	258	1-,2-,4-, and 8-bit data for gray and Indexed images; 8-bit for RGB, CMYK, CIELab, HiFi.
Compression	259	1 (uncompressed), 2 (basic CCITT 1D), 5 (LZW), 32773 (PackBits), 32895 (IT8 uncompressed), 32896 (IT8 linework compression)
PhotometricInterp	262	0 (monochrome), 1 (monochrome), 2 (RGB), 3 (RGB Palette), 5 (CMYK and hifi/multi-ink), 8 (CIELab), 9 (ICC CIELab).
StripOffsets	273	StripOffsets or TileOffsets is required. No limit on number of strips.
SamplesPerPixel	277	1 (monochrome and Indexed), 3 (RGB, CIELab), or 4 (CMYK). Variable for multi-ink/hifi images. Value may be larger if ExtraSamples are present.
RowsPerStrip	278	Anything between 1 and ImageLength, inclusive.
StripByteCounts	279	StripByteCounts or TileByteCounts is required. No limit on number of strips.
XResolution	282	All valid values are supported. If not present, default is taken to be 72dpi.
YResolution	283	" "
PlanarConfig	284	1 (Chunky) preferred for RGB, CIELab. 2 (Planar) preferred for CMYK, multi-ink/hifi.
ResolutionUnit	296	1 (no unit), 2 (inch), and 3 (centimeter) are allowed.
Predictor	317	1 (no predictor) and 2 (horizontal differencing) are supported for LZW compression.
ColorMap	320	Required for RGB Palette Color images
TileWidth	322	Required for tiled images
TileLength	323	Required for tiled images
TileOffsets	324	Required for tiled images
TileByteCounts	325	Required for tiled images

SubIFDs	330	See the TIFF Trees section.
InkSet	332	Both 1 (CMYK) and 2 (multi-ink/hifi) are supported.
InkNames	333	Required if InkSet = 2.
NumberOfInks	334	Required if InkSet = 2.
DotRange	336	Supported.
ExtraSamples	338	Supported, including support for associated and unassociated alpha data.
ClipPath	343	See the TIFF ClipPath section.
XClipPathUnits	344	" "
YClipPathUnits	345	" "
Indexed	346	See the Indexed images section.
OPIProxy	351	See the OPI Proxy image section.
ImageID	32781	Written out as %%MainImage OPI comment.
IT8RasterPadding	34019	See the ANSI IT8 TIFF/IT specification
IT8ColorTable	34022	" "

Other information	
Defaults	Fields not present in the TIFF file will automatically be given their default values, as required by the TIFF specification.
	Tags not listed above are ignored.
Byte Order	Both MM- and II-style TIFF files are supported in PageMaker, in both Apple Macintosh and Microsoft Windows versions. There is no need to convert files from one form to another when transferring them to the other platform.
OPI comments	In addition to the standard OPI 2.0 comments, all ASCII TIFF fields are copied to the PostScript language output stream as OPI comments.
Size of integers	PageMaker supports BYTE, SHORT, and LONG for all integer TIFF fields
Filenames and filetypes	PageMaker does not require that Mac TIFF files are of type TIFF, as long as the filename ends in ".tif."