Table of Contents

Journal feedback: suggestions for articles, questions, etc.

Selecting Paper Trays in PostScript

You want to print your document on the three-hole-punch paper you just loaded into your printer's top tray. How do you tell your PostScript printer to draw paper from this tray? Well, you can't, directly.

This is such a common task, it continues to astonish me that there's no way for a PostScript program to specify that paper should be drawn from a particular tray.

So, again, how do you tell your PostScript printer to draw paper from the top tray?

In a word: indirectly.

Let's see how.

setpagedevice

The obvious PostScript operator to be in charge of tray selection is *setpagedevice*. This is the operator that specifies the use of printer-specific features, such as duplex printing, paper size, and manual feed.

What we wish for is a key-value pair that would tell *setpagedevice* to draw paper from a specific tray, something like:

```
<< /ActiveTray 0 >> setpagedevice
```

Unfortunately, there isn't any such key-value pair defined in PostScript.

Happily, we can create similar functionality by hijacking the *MediaType* mechanism.

Hijacking *MediaType*

One of the printing parameters you *can* specify in PostScript is the type of media on which the current job should be printed. You do this by handing the name of the media type to *setpagedevice:*

```
<< /MediaType (Three-Hole Punched) >> setpagedevice
```

This tells the PostScript device to draw paper from whichever tray contains the three-hole-punched paper.

The obvious question, of course, is "How in tarnation does the PostScript interpreter know which paper tray contains the punched paper?"

The answer is: we tell it.

Among the key-value pairs that have meaning to *setpagedevice* is *InputAttributes.*

```
<<   /InputAttributes  <<
          0 << /PageSize [ 612 792 ]
               /MediaType (Three-Hole Punched)
            >>
          1 << /MediaType (Letterhead) >>
     >>
>> setpagedevice
```

The *InputAttributes* entry does not request a printer feature; rather, it asserts things that are true of one or more paper trays on the current device. The value of *InputAttributes* is a dictionary containing one entry for each tray whose attributes you are specifying.

Each entry consists of a tray number (trays are numbered starting at zero) and an associated dictionary. Each key-value pair within a tray dictionary asserts something that is true of that tray. For example, the code snippet above asserts that Tray 0 contains three-hole-punched, American-letter-size paper (612 x 792 points is 8½ x 11 inches).

The Wonders of MediaType

Of importance to our topic is the fact that *InputAttributes* can specify a MediaType for a particular paper tray. The MediaType value is an arbitrary string. When a call to *setpagedevice* asks for a particular MediaType

```
<< /MediaType (Three-Hole Punched) >> setpagedevice
```

that string is compared to the MediaType strings stored for each paper tray; if it finds a match, then that is the paper tray that is used.

So, the way to specify paper trays is to initially associate a particular MediaType string with each tray and then later use the appropriate MediaType to choose the desire tray.

Thus, your PostScript code could start with:

```
<<   /InputAttributes  <<
          0    << /MediaType (UpperTray) >>
          1    << /MediaType (MiddleTray) >>
          2    << /MediaType (LowerTray) >>
     >>
>> setpagedevice
```

This associates a descriptive MediaType string with each paper tray. Trays 0, 1, and 2 now contain MediaTypes (UpperTray), (MiddleTray), and (LowerTray).

It is now simple to specify, at the beginning of a page in your PostScript code, the tray that page should use as a paper source. If a page needs to draw its paper from the middle tray, then

```
<< /MediaType (MiddleTray) >> setpagedevice
```

should do the trick.

The Downside? There isn't any particular disadvantage to this, aside from its being a bit clunky. However, since this is the only way of selecting a particular tray, the clunkiness is irrelevant.

The Caveat As is true of everything associated with *setpagedevice,* remember that printer features vary from one device to the next. In our case, remember that not all printers have multiple paper trays. Among those printers that *do* have two or more trays, the assignment of tray numbers to physical paper trays is printer-specific. Tray 0 may *not* be the upper tray on a particular printer.

That being the case, explicitly specifying a paper tray should be reserved for PostScript code that will be executed on a known printer (perhaps the in-house production printer, for example).

Touching Up Images in Acrobat 9

Among Acrobat's tools that I rarely use, but when I need them, I *really* need them are the Touch-up Text and Touch-up Object tools. The former allows you to make minor changes to text within a document: correct spelling, change a phone number, etc. The latter allows you to make a variety of changes to everything *other* than text in your document.

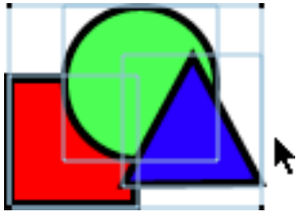
I only recently had occasion to use the Touch-Up Object tool on an image in one of my PDF files and I was surprised to find how powerful the tool is. You can rotate, crop, flip, and even replace an image on the page without ever leaving Acrobat.

In this *Journal* article, we'll take a look at just how you do this.

The Touch-Up Object Tool

The Touch-Up Object tool is at the right end of the Advanced Editing toolbar, as at right.

When you select this tool, the mouse pointer turns into a standard leftward-pointing arrow. With this pointer, you can click on anything on a PDF page: an image, line art, text; the object will now be selected, with a rectangular border and standard "handles" at each corner. (You can even shift-click to select multiple objects; the selection-rectangle-with-handles will enclose the entire collection of items you selected, as at right.



Having selected an object, you can then perform a variety of actions on that object. The exact set of possibilities depends on the type of object you have selected: text block, line art, or image. (Note that if you have selected a text block, you cannot edit the text; you can only move about or otherwise modify the block of text as a whole.)

Common Actions

No matter what type of object you have selected, there are some things you can always do:

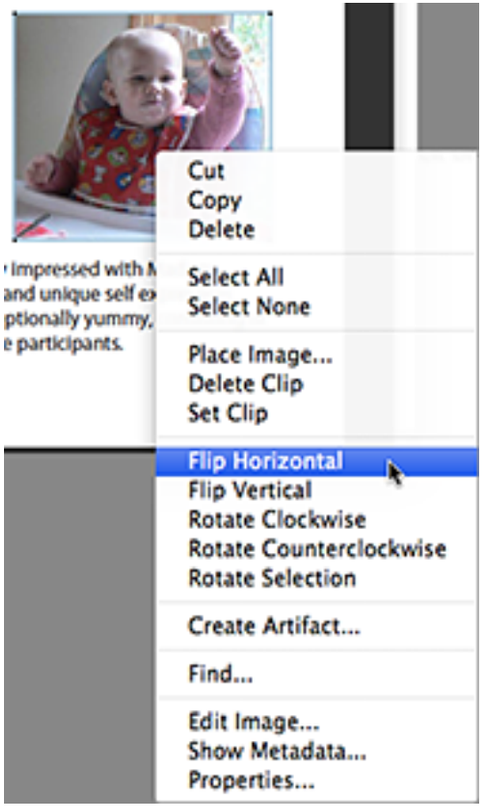
- You can drag the object around the page by clicking in the middle of the selected object and dragging it to the desired location.
- You can resize the object by clicking and dragging one of the corner handles.

Less-Common Actions

If you right-click on the selected object, you will get a pop-up menu of additional actions you can take with that object. The contents of this menu are dependent upon the type of object; in this article, we'll look at the changes you can apply specifically to images.

Touching Up Images

When you right-click on an image with the Touch-Up Object tool, you obtain the pop-up menu shown at right.



Many of the items in this menu are fairly straightforward; Cut, Copy, Delete, Flip, and Rotate all have their intuitive meanings.

A few of them are less obvious; we'll discuss these.

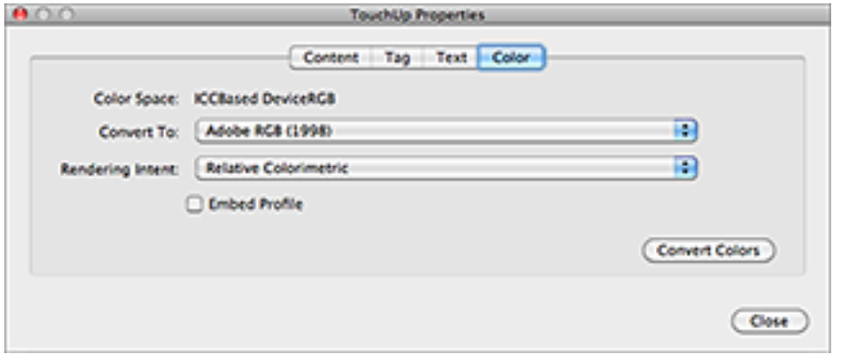
One-Liners

Three of these menu items are very easy to discuss; what they do can be described in a single sentence (using semicolons, perhaps).

Create Artifact Tags the image as an *artifact,* something that should be ignored by the Acrobat "Read-Aloud" feature.

Show Metadata Brings up a dialog box that displays whatever metadata may be embedded in the image, including such things as camera make and model.

Properties Brings up a dialog box that displays the properties of the image, including colorspace; you can also convert the image to another colorspace, if you wish.

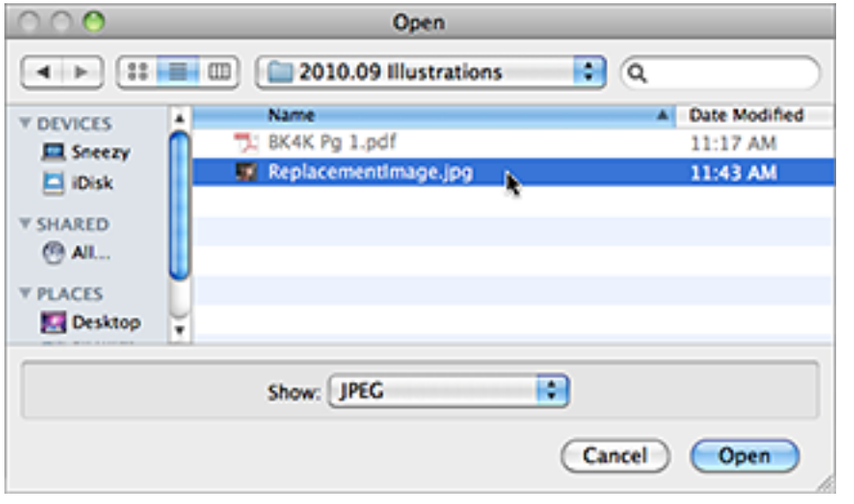


Multi-Liners These features are easy to understand and use; they just can't be described in a single sentence.

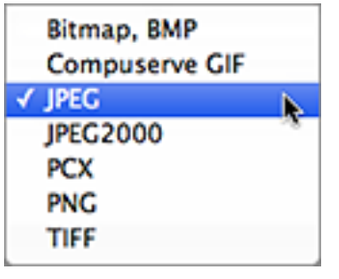
Edit Image Now, this one's cool: Acrobat opens the image up in the image editor of your choice (Photoshop, in my case). You make whatever changes your heart desires and then, when you save the image, the editor saves the modified image back to Acrobat. The new version of the image appears on the PDF page.

One of my favorite Acrobat features!

Place Image This command lets you replace the selected image with another image chosen from your hard disk. When you select this item, Acrobat presents you with a standard pick-a-file dialog box (right). Just select the image file you want to use and Acrobat will place it on the PDF page, removing the original image.



One minor peculiarity about this dialog box: it enables the selection of only one type of image at a time (on the Mac, anyway; I haven't checked Windows). This means that you must explicitly choose the type of image you want to select from the pop-up menu. Why Adobe didn't just enable all supported image formats, I don't know.



Set/Delete Clip

As I said earlier, when you select an object with the Touch-Up Object tool, the object is surrounded by a bounding rectangle that has a little square handle at each corner. Usually, when you drag these corners, Acrobat resizes the image, yielding a larger or smaller version of the same image.

If you select *Set Clip,* Acrobat changes the mouse pointer and then waits for you to drag one of the image's handles. When you do so, rather than scaling the image according to your mouse drag, Acrobat clips the image to the final rectangular region defined by your mouse drag.

The *Delete Clip* command releases the clipped region, making the entire image visible again.

Appropriate Use

PDF files should always be created with images properly sized, rotated, and positioned. When something is wrong with an image on a PDF page, the proper thing to do is to go back to the original source document and fix the problem in InDesign, Word, QuarkXpress, or whatever.

However, sometimes circumstances don't allow us that luxury. The deadline is *right now!* I must get the PDF file to the client! No time to fix the original document and there's only a slight tweak needed to the image. Ah! The Acrobat Touch-Up Tools! Thank you, Adobe.

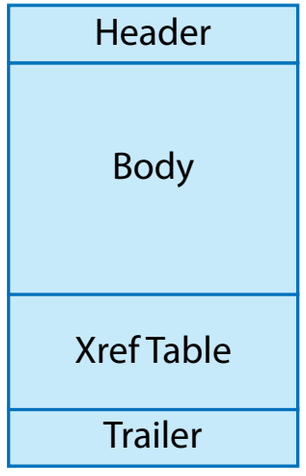
Of course, eventually you should return to the source document and fix the problem for good. But as a band-aid applied to a minor problem, the Touch-Up tools can't be beat!

PDF & XPS Nuggets

PDF Nugget

PDF File Structure

The first time you open a PDF file in a text editor, you are faced with a sea of unfamiliar code, much of it compressed and unreadable. At the highest level, a PDF file has a simple structure, consisting of four parts:



Header — This is the first line in the PDF file; it consists of a snip of text looking like "%PDF-1.7", indicating the version of the PDF spec this file follows.

Body — This is the bulk of the PDF file, consisting of everything needed for the PDF document; this is where the pages, drawing commands, image data, and everything else making up the document contents.

Cross-Reference Table — This table identifies the location—as an offset from the file's start—of every PDF object within the file. (A PDF object is a numbered representation of a single piece of data (font definition, image, etc.) within the file. The table starts with the keyword *xref*.

Trailer — This ending part of the file identifies the location of the cross-reference table and other "bootstrap" information for the file. The PDF Trailer is actually the first part of the file inspected by a PDF viewer when you open the file. It starts with the *Trailer* keyword.

The above organization is required of every PDF file, regardless of the software that created it.

The only complication to this picture is that there may actually be two or more body-xref-trailer sets in a PDF file, a consequence of something called "incremental updating."

But, that's for another nugget.

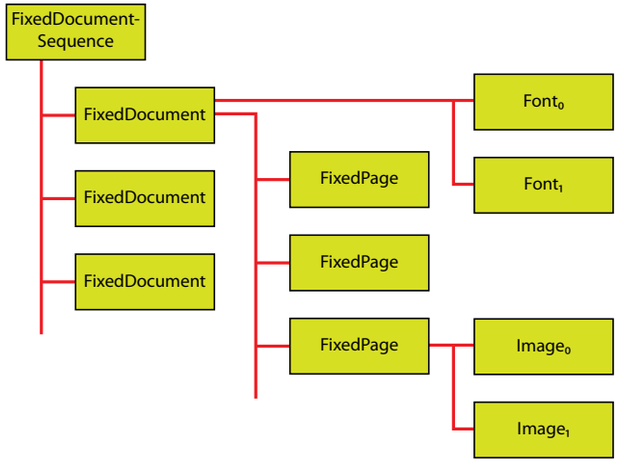
XPS Nugget:
XPS Parts

As we discussed in the previous issue, an XPS file has a hierarchical structure that may be thought of as consisting of internal files residing in folders, all within the zip-compressed package that is the XPS file. The component files within the XPS file are referred to as "parts" in XPS terminology; each provides some piece needed for the entire XPS package.

There are several types of XPS parts; chief among these are:

FixedDocumentSequence

An XPS package may contain several documents; a *FixedDocumentSequence* part specifies the location within the package of the individual documents contained in the XPS file.



FixedDocument This part specifies the characteristics of one document within the XPS file and the locations of the pages within the package hierarchy.

FixedPage This part contains the drawing commands that paint the contents of an individual page.

Font Not surprisingly, this defines a font used in the package. XPS supports TrueType and OpenType fonts. (Not PostScript Type 1, note.)

Image This part contains an image used in the package. Images in XPS may be any of the following formats:

- JPEG
- TIFF
- PNG
- Windows Media Photo

Many other types of parts are defined for XPS files, but the above brief list should give you a good idea of the broad concept.

Schedule of Classes, October 2010–January 2011

Slim Pickin's

I won't be teaching many classes this coming quarter. The Book will be occupying all my time.

At right are the dates of Acumen Training's upcoming classes. Clicking on a class name will take you to the description of that class on the Acumen Training website.

O.C. and On-Site

These classes are taught in Orange County, California and on-site at corporate sites world-wide.

Please see the Acumen Training web site for more information, including an up-to-date schedule.

Class Fee

Classes cost $2,000 per student, with the following exceptions:

- XPS class $1,500
- *Troubleshooting PostScript* $1,500
- *Support Engineers' PDF* $1,000

PDF Classes

PDF 1: File Content and Structure			Jan 10–13
PDF 2: Advanced File Content			
Support Engineers' PDF		Dec 9–10	

PostScript Classes

PostScript Foundations	Sep 20–24		Jan 17–21
Advanced PostScript			
Variable Data PostScript			
Troubleshooting PostScript		Dec 6–8	

XPS Classes

XPS File Content and Structure		Dec 13–15	

There is a 10% discount for signing up three or more students.

Note that if you have four or more students that need to take a class, it will almost certainly be cheaper to arrange an on-site class.

Contacting John Deubert at Acumen Training

For more information

For class descriptions, on-site arrangements or any other information about Acumen's classes:

Web site: www.acumentraining.com **email:** john@acumentraining.com

telephone: 949-248-1241

mail: 24996 Danamaple, Dana Point, CA 92629

Registering for Classes

To register for an Acumen Training class, contact John any of the following ways:

Register On-line: www.acumentraining.com/register.html

email: john@acumentraining.com

telephone: 949-248-1241

mail: 24996 Danamaple, Dana Point, CA 92629

On-Site Classes

Information regarding classes on corporate sites is available at www.acumentraining.com/Onsite.html. These courses are taught throughout the world; for additional information on classes outside the United States, go to www.acumentraining.com/OnsitesWorldWide.html.

Back issues

All issues of the *Acumen Journal* are available at the Acumen Training website: www.acumenjournal.com/AcumenJournal.html

What's New at Acumen Training?

New Book a-Comin'

You'll notice on the Schedule page that I'm not teaching many classes the rest of this year. I'm working on a new book, which will occupy most of my time through November. (Which then leave us staring at the end-of-year-holidays season; lovely time of year, but useless for scheduling classes.)

As always, if you want to see the latest schedule of classes, go to

www.acumentraining.com/schedule.html.