John Deubert's **Acumen Journal, March 2009**

Table of Contents

The Acrobat User

Acrobat 9's *Split File* Feature

Acrobat 9 can automatically split a PDF file into a series of smaller files according to size, page count, or chapters. It's easy.

Short Articles This Time

The *Journal* articles are a bit shorter than usual this time. Next time, too, probably.

I'm putting as much effort as I can into getting the XPS class ready for May 5.

PostScript Tech

PostScript, PDF, & XPS

The Acumen Training XPS class is nearly finished and after months of assembling student notes, overheads, and handouts, I thought it would be interesting to compare it to its older cousins, PostScript and PDF.

Class Schedule

March, April, May, June

What's New?

XPS course description & class outline

There's now a day-by-day class outline available for the new XPS course. The first class is May 5.

Contacting Acumen

Telephone number, email address, postal address

Journal feedback: suggestions for articles, questions, etc.

PostScript, PDF, and XPS

I am in the final stages of assembling a course in XPS—*XML-based Paper Specification*—a document format introduced by Microsoft at the beginning of the Windows Vista era. It's been an interesting undertaking, in large part because all of my other classes are based on Adobe technology and software: PostScript and PDF share an underlying graphic model and so have many similarities in how they approach drawing line art and text.

Not so, XPS. its file format and graphical underpinnings are very different from the Adobe formats. Since Microsoft intends XPS to be a replacement for PDF as a means of document distribution and printing, I thought it would be interesting to compare some of the characteristics of XPS, PDF, and PostScript.

This article is pretty short, since I'm up to my ears working on the new class notes. It also isn't anything like a complete comparison of all the characteristics of the three document formats. Still, it will give you a start toward understanding how the three languages compare.

For more information, take the XPS class. (Bring the kids! Make a vacation out of it!)

Intent & Implementation

Let’s start by considering the intended purpose of each of the three file formats and how each was designed to fill that purpose.

PostScript

PostScript is, at root, a printing language; its purpose has always been to describe (to a printing device) the pages within a document.

But, then, you know this

Readers of *PostScript Tech* already know PostScript (you *have* taken a PostScript class, haven’t you?), so the PostScript sections of this discussion are presumably review.

To make PostScript maximally capable, Adobe made it a programming language; a PostScript stream is an executable program sent to an interpreter within the printer. The interpreter executes the program, renders the result into an internal bitmap, and transfers the bitmap to paper.

Programming has always been PostScript’s glory and its hobble; PostScript can describe any kind of document, however large, complex, or exotic. However, PostScript programs can vary widely in efficiency, size, and execution speed, according to the programming skills of whoever produces the stream. Programming languages always provide a myriad ways of carrying out a given task, most of them wrong; as a result, some PostScript programs will be compact and fast, while others, producing the same visual result, will be large, bloated, and slow.

PDF

PDF is intended as a document description language. Adobe applied its 10 years of experience with PostScript to create a file format that was leaner, faster, and more compact than the earlier language. As with PostScript, Adobe intended to make it possible to express *any* document as a PDF file. Originally, this meant words, graphics, and images on paper. Over the years of PDF’s existence this has grown to include movies, sounds, and other multimedia; interactive forms for collecting and reporting data from users; digitally-signed contracts and other papers of legal consquence. Today, Adobe’s goal is still to make PDF capable of embodying any type of document, but it now interprets “document” in the broadest possible sense.

Unlike PostScript, PDF is not a programming language; the number of ways you can produce a given page is very much smaller than in PostScript. As a result, it is much easier to produce an efficient, compact page description in PDF.

XPS

XPS, like PDF, is intended to describe documents. However, Microsoft's sights are as yet much more restricted than Adobe's has become with PDF and that's not a bad thing. XPS is a format for distributing, viewing, and printing documents whose pages contain line art, text, and images. Period. No form fields, no movies, no annotations; just marks on a page. Within this scope, XPS has capabilities equivalent of PDF. Because the intent is limited to marks on a page, XPS is easier to parse than PDF and, once you know how to get to its contents, it is human readable.

We'll talk about the "get to its contents" part in a moment.

Graphic Model

PostScript

PostScript consciously adopted a traditional graphic artist's view of the universe. In drawing a graphic figure in PostScript, you supply drawing commands that describe how you would produce that figure using pen on paper. To draw a blue-filled, 2-inch-wide-and-high triangle in PostScript you would use code similar to the following:

```
0 0 1 setrgbcolor
144 288 moveto  288 288 lineto  216 432 lineto   closepath
fill
```

Pick blue ink. Move to a place on the page; draw a line, draw a line, close the figure; fill it with ink. The units are 1⁄72-inch "points," close to (or exactly equal to, depending on how old you are) the long-standing graphic artists' unit.

PDF PDF is a descendant of PostScript; it inherits from the earlier format the assumptions about how to describe a graphic object. Units are the same 1⁄72-inch and graphics are described as a series of drawing instructions. Our blue triangle would look like this:

```
stream
0 0 1 rg
144 288 m   288 288 l   216 432 l   h
f
endstream
```

Drawing commands in PDF are encapsulated in a *stream,* which represents a series of bytes that must be interpreted sequentially (as opposed to most of the PDF file, which can be randomly accessed). The drawing commands themselves are functionally the same as in the PostScript triangle, using more-compact names for the drawing commands (*m* for *moveto,* etc.). The PDF vocabulary diverges from PostScript's as you move to more elaborate graphics, but it is always evident that the one derived from the other.

XPS Where Postscript and PDF describe a page in terms of the drawing commands that put marks on the page, XPS describes the graphic objects that reside on that page.

To describe the blue triangle, both PostScript and PDF say: "Start here; draw a line to here; now draw a line to here; close the figure; select blue ink; fill the graphic"

XPS says, "There is a polygon on the page; it starts here; it has vertices at the following locations; it is closed; it is filled with blue."

The XPS code looks like this:

```
<Path Fill="#0000ff">
    <Path.Data>
        <PathGeometry>
            <PathFigure StartPoint="192,672" IsClosed="true">
                <PolyLineSegment Points="384,672 288,480" />
            </PathFigure>
        </PathGeometry>

    </Path.Data>

</Path>
```

Some differences are immediately evident when you examine this code:

- Since XPS is an XML language, the triangular path is packaged in a series of XML elements.
- The XPS coordinate system is different from PostScript/PDF; the origin is at the upper-left corner of the page and units are 1⁄96-inch, matching Windows' standard graphics coordinate system.
- XPS can be pretty verbose; the above code has more than triple the character count of the PDF snippet. This is eased by the existence of an…

Abbreviated Format

To ease the large amount of text required to describe such routine things as graphical objects, XPS has an abbreviated format for constructing paths. Here's our triangle again:

```
<Path Fill="#0000ff"
    Data="M 192,672  L 384,672 288,480 Z"
/>
```

This is *much* better; we have a series of *moveto* and *lineto* drawing commands that are a fraction the size of the original version. I'm not clear why this format isn't universally used for line art, but looking over commercially-produced XPS code, I find the verbose style is very common.

Go figure.

Bytes on Disk

PDF, XPS, and PostScript all describe documents for the benefit of viewers and printers. How are these document descriptions packaged on disk?

PostScript

A PostScript file contains a single stream of executable PostScript code, a program that describes a series of pages. PostScript code is primarily intended to be used as a print stream, so it is designed to be consumed sequentially and incrementally. Usually, the stream starts out with a series of procedure definitions, followed by the code that draws page contents, one page at a time in page order.

```
%!PS-Adobe-3.1
/bd { bind def } bind def
/ld { load def } bd
...
... lots more definitions
...

... Page 1's
... drawing commands
...

... Page 2's
... drawing commands
...
```

PDF

A PDF file is a bit more complicated, but still conceptually straightforward with regard to simple page description. A PDF file defines a series of “objects,” each of which encapsulates a single piece of data that is used in the document definition: an image, a set of drawing commands, a font definition, etc.

If you open a PDF file in a text editor, you are likely to see a series of blocks that look something like this:

```
%!PDF-1.8

100 0 obj
...
...
endobj

120 0 obj
...
...
endobj

80 0 obj
...
...
endobj
```

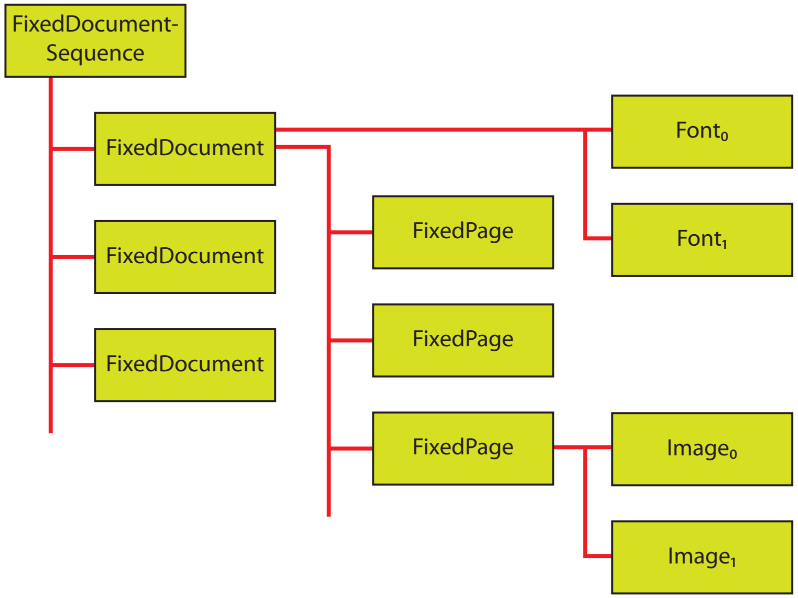
```
120 0 obj
<<
     /Catalog true
     ...
>>
endobj
```

A quite large fraction of the file will be compressed and so not readable to the unclothed eye. This is good, since it reduces the size of the file, at the cost of a minor loss of portability (since the file is no longer straight ASCII).

XPS

An XPS file is actually a zip file with a *.xps* suffix. This compressed package contains a potentially large number of component files, each supplying something needed by the document embodied by the XPS file: an image, a font, a page contents, etc. To an extent, items that are objects in a PDF file are individual files within the XPS zip package.

Some of the files within the XPS package exist to specify relationships among the other components. Thus, a relationship file might specify that the component file */Document/1/pageContents/Page1.fpage* uses a font stored in the component file */Resources/Fonts/arial.ttf.*



The components of the zip-compressed XPS package generally occupy a hierarchical structure reflecting the purposes of its components; a typical layout is diagrammed at right.

Multiple Documents

An XPS file can contain multiple documents. Thus, XPS can be used to distribute a whole series of documents in a single package.

Acrobat added this ability to PDF with the introduction of *PDF portfolios.* The capabilities of portfolios are extremely broad, including support for title pages, presentation, etc. See the November 2008 issue of the *Journal* for more information.

A PostScript stream draws a series of individual pages. PostScript neither knows nor cares how many documents the pages represent.

You can examine the contents of an XPS package by simply changing its suffix from *.xps* to *.zip* and then handing the renamed file to your favorite *zip* utility. The result will be the XPS file's contents placed as a set of files and folders on your computer.

Other Comments

PostScript

PostScript is the only one of the three formats that was intended to be primarily a print stream. It is strictly sequential in its organization and may be consumed by a printer incrementally, eliminating any need for buffering the entire file. PDF and XPS, on the other hand, require random-access of the file contents; using either type of file as a print stream requires that the printer have a hard disk or sufficient RAM to store the entire file.

PostScript is also a programming language and contains a huge set of capabilities that makes it an excellent choice for doing such tasks as variable data printing. Also, it's fun to mess around with by hand (*yes,* it is!), whereas the other two formats are not intended to be hand edited, let alone hand constructed.

PDF

Graphically, PDF has much the flavor of a re-designed PostScript. Though it implements the same graphical model as PostScript, PDF is not a programming language and therefore is much more straightforward in its page descriptions; it lacks the conditional execution, procedures, loops, and other constructs that make PostScript a more difficult language to parse.

In capability, PDF has grown dramatically since it was first introduced in 1994. Adobe's original PDF concept was of a simple file format embodying an unchangeable document for electronic distribution and display. Over its 15 years of existence, the PDF has acquired support for an remarkably large body of features, including interactive form fields, digital signatures, annotation and review, and much more; so much more, in fact, that these days it's hard to state in a simple sentence just what the PDF file format is for.

PDF has by far the broadest capabilities of the three file formats, but many people consider it to have become bloated and no longer the clean, simple document format of their youths. My own feeling is that if your document doesn't need the vast accumulation of PDF file features, then don't use them; if you limit yourself to the simple description of a document, PDF is still a clean, pleasing file format.

XPS

XPS implements the original PDF concept: an XPS package does nothing except describe the appearance of a document. It can indicate clickable links within the document and that is pretty much the only special effect: no movies, no form fields, no annotations.

If you have need of form fields or animation, you will need to pick a different file format. On the other hand, the vast majority of distributed documents are simple ink-on-paper and for these, XPS provides a compact format free of extraneous slush whose only purpose (from a strict marks-on-paper standpoint) is to take up space and bandwidth.

One thing that XPS does have that PDF does not—and this is terribly important for a printed document—is support for a job ticket. The XPS file can contain information on paper type, duplexity (if "duplexity" isn't a word, it should be), and other details of how the document should be printed. Printer engineers have been wishing for years that Adobe would add this to PDF.

Which is Best?

Actually, they are different enough that I don't think that question can be meaningfully answered. Still, some commentary is possible:

PostScript?

From the standpoint of a printer engineer, PostScript is the best of the formats, because it was designed from the start to be a print stream. It can be consumed and interpreted incrementally, which reduces the memory and disk storage requirements of a printer. However, as a document distribution format, PostScript is probably the least good, since a PostScript document tends to be larger than the PDF or XPS equivalents.

Besides printing, PostScript is at is best for people doing variable data printing and other tasks that can use PostScript's programming language personna; many people have implemented variable data printing systems build around handwritten PostScript code. (If you need to do this, you should look at the Acumen Training Variable Data PostScript class.)

PDF? If you are distributing a document with form fields, sound, animation, or other interactive elements, then PDF is the only candidate among the three we are examining here. Also, the PDF universe is where you *must* be for annotating, reviewing, and otherwise sending documents around for commentary. I like PDF a lot for all the things it does besides specify marks on paper.

Also, PDF viewers are ubiquitous; you can distribute a document in PDF format and be confident that people will be able to read it.

XPS? XPS has yet to grow into its potential. Although it's a very good format for distributing documents, only Windows Vista and (presumably) Windows 7 have native support for viewing XPS documents. Microsoft makes available utilities for viewing XPS files on Windows XP AND 2000 (go here) and there are third-party tools for viewing them on the Mac, but the people to whom you send the document will have to have installed one of those utilities. It's not a big deal, but I always prefer to not make my customers do any work; they're likely to lose interest and go do something else.

I expect this will change. Microsoft seems to be serious about XPS and over time more people will perforce migrate to Vista or Windows 7. I expect that Microsoft will eventually release Macintosh and Linux viewers, since they hope to make XPS an industry standard.

I rather hope they do. I think there's room for a document format that is streamlined solely to the task of distributing documents.

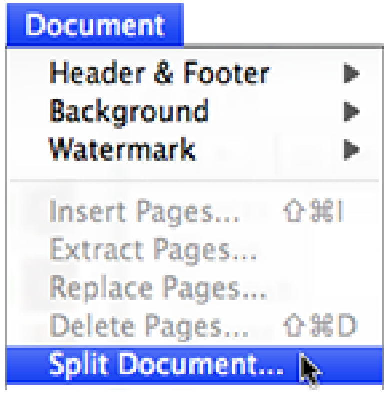
XPS Viewers

There are at least a couple of Mac viewers with which I have some experience:

- *NiXPS* has a $99.00 viewer that works very well. It's my favorite on the Mac. (I've been using it to develop the Acumen Training XPS class.)
- *SanaTech's* $100.00 viewer works well enough, though it's much rougher around the edges. Also, the company is absolutely unresponsive to support questions.

Linux support is spotty at this point. SanaTech's viewer is available for Linux and Artifex Software is working on GhostXPS.

The *Split Document* Feature



I often have to wade through language specifications and other tomes distributed as PDF files. These documents are often very long; for example, the XPS specification, which I am getting to know depressingly well, runs to over 400 pages.

I prefer to read this kind of document on paper; electronic documentation is fine for reference or short documents, but when learning something that is new to me and complex, I like to have printed paper documentation, usually spread out all over my office floor.

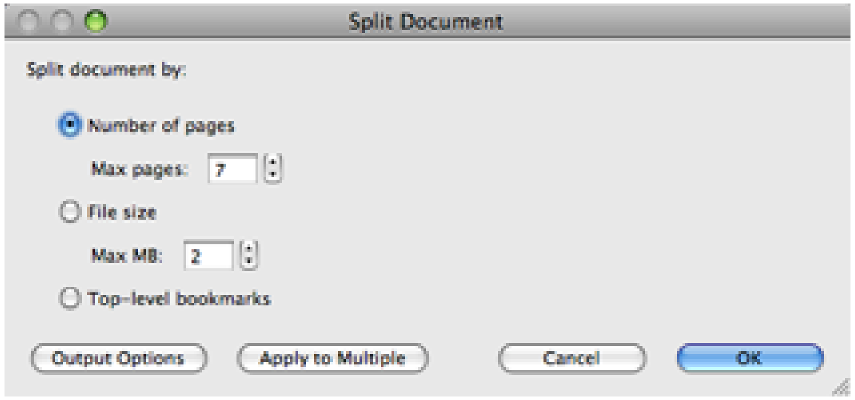
I usually want to print these large documents in conveniently-sized chunks. Acrobat 9 makes this remarkably easy with a new *Split Document* feature, located on the *Document* menu.

This feature splits the current PDF document in chunks of equal page count, equal number of bytes, or according to the document's bookmarks.

I find myself using it all the time.

Let's look at it.

It’s Pretty Darned Easy



Split Document

Split document by:

● Number of pages

Max pages: 7

○ File size

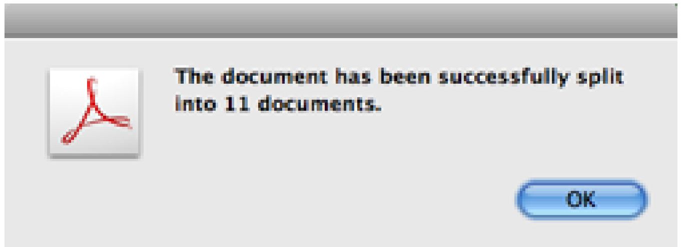
Max MB: 2

○ Top-level bookmarks

Select *Document>Split Document* and Acrobat will present you with the dialog box at right. Here you specify the criterion that the software should use to decide where to split your document. You have your choice of:

- Split every *n* pages; all of the resulting fragments will have the same number of pages except, probably, the final one.
- Split the document into pieces that are not more than a specified size on the disk. The resulting fragments will be more-or-less the same size, but none greater than the maximum you specify.
- Split at the top-level bookmarks. This is the one I usually choose for specifications and the like, since the split turns each chapter into a separate PDF file.

There are some options you can specify by clicking the *Output Options* button, but we’ll come back to those in a moment.



The document has been successfully split into 11 documents.

When you click the *OK* button, Acrobat will think to itself for a few seconds (maybe a minute if it’s a long document being split into lots of sections) and then display a dialog box telling you how many new PDF files it created.

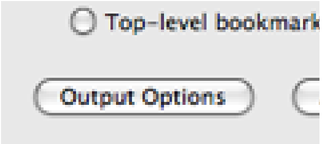
That’s it.

Easy, huh?

Output Options

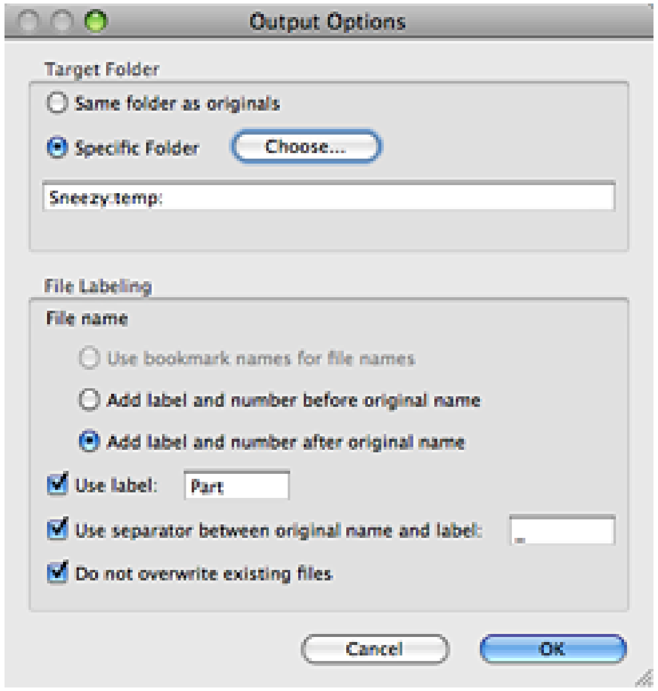
The Split Documents feature lets you specify the location and naming of the fragments it generates.

If you click on the *Output Options* button in the *Split Document* dialog box, Acrobat presents you with the dialog box at lower right. There are two set of controls here.



Target Folder

You can have the fragment files stored in the same folder as the original file or have them placed in a folder of your choosing. The two radio buttons and the *Choose* button are pretty standard.



Output Options

Target Folder

○ Same folder as originals

● Specific Folder Choose...

Sneezy:temp:

File Labeling

File name

○ Use bookmark names for file names

○ Add label and number before original name

● Add label and number after original name

☑ Use label: Part

☑ Use separator between original name and label: _

☑ Do not overwrite existing files

Cancel OK

File Labeling

Here you specify how the fragment files should be named. You can choose between two naming strategies:

- Add a label and sequence number to the original name. (The additions can be added before or after the original.)
- If you are splitting the documents at the top-level bookmarks, you can use those bookmark names for the fragments.

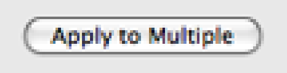
There are also checkboxes and text fields that let you specify:

- What the label should be.
- Whether you want a separator between the label and the original filename and what that should be.

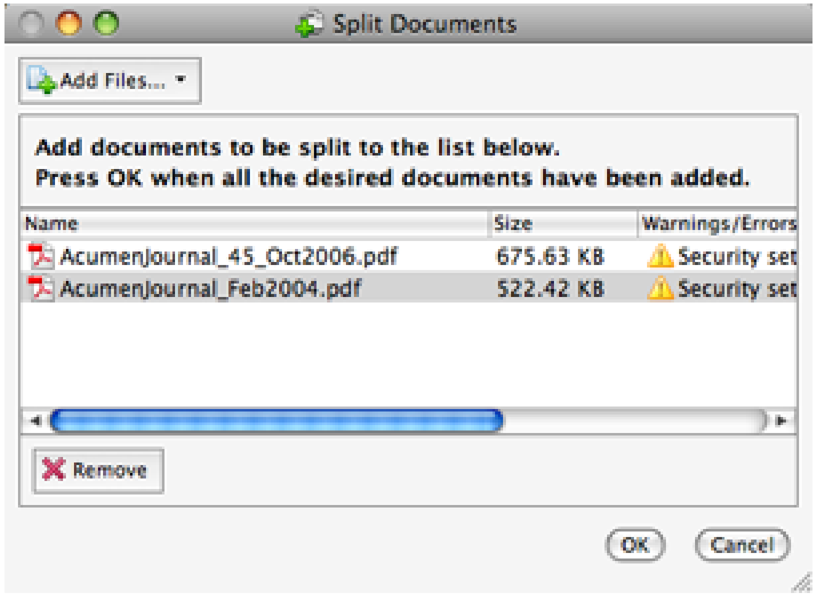
The default label is “Part” and the default separator is “_”, so splitting a PDF file named “Paw Prince.pdf” would yield fragments named *Paw Prince_Part 1.pdf, Paw Prince_Part 2.pdf,* and so forth.

Finally, you can decide whether Acrobat should overwrite an existing file whose name happens to match that of a fragment. Interestingly, this checkbox doesn’t seem to have any effect; Acrobat always overwrites the existing file.

Multiple Files



Finally, if you click the *Apply to Multiple* button in the *Split Documents* dialog box, Acrobat present you with a dialog box that lets you select several files that should be split up. This dialog box is easy to figure out, so I'll dispense with a detailed description.



Add documents to be split to the list below.
Press OK when all the desired documents have been added.

Name	Size	Warnings/Errors
AcumenJournal_45_Oct2006.pdf	675.63 KB	Security set
AcumenJournal_Feb2004.pdf	522.42 KB	Security set

Easy, Isn't It?

That's all there is to tell about this feature. It's easy to use, does what it says it will, and I find it periodically indispensable.

What more can one ask of a feature?

Schedule of Classes, March – June 2009

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.

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.

PDF Classes

PDF 1: File Content and Structure	Apr 6–9		Jun 22–25
PDF 2: Advanced File Content			
Support Engineers' PDF	Apr 23–24		Jun 18–19

PostScript Classes

PostScript Foundations	Mar 23–27	May 11–15	
Advanced PostScript	Mar 30–Apr 2		
Variable Data PostScript			
Troubleshooting PostScript	Apr 20–22		Jun 15–17

XPS Classes *(New!)*

XPS File Content and Structure		May 5–7	Jun 10–12

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

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?

XPS Class Outline Now Available

A course description, including a day-by-day outline, of the *XPS File Contents and Structure* class is now available here on the Acumen Training website. The first class is now scheduled for May 5 at Acumen Training's classroom site in Costa Mesa, California (near the Santa Ana/John Wayne airport). The three-day class will present in-depth coverage of the essentials of the XML-based Paper Specification.

See the website for the detail description, but the class includes such topics as:

- Open Packaging Convention
- XPS file structure
- Line Art
- Coordinate system
- Color
- Text
- Font Support
- Images
- Pattern fills
- Clipping
- Drawing curves

Like all Acumen Training course, *XPS File Structure and Contents* concentrates on those parts of the file format that apply to putting marks on a page. It is intended for

- Printer engineers working on devices that must consume XPS as a print stream
- Support engineers who support those devices
- Software engineers generating XPS for printed or displayed documents.

Don't hesitate to contact me to ask questions (or arrange an on-site).

I hope to see you there!

Journal Feedback

If you have any comments regarding the *Acumen Journal,* please let me know. In particular, I am looking for three types of information:

Comments on usefulness. Does the Journal provide you with worthwhile information? Was it well written and understandable? Do you like it, hate it? Did it cause the fillings to melt out of your teeth?

Suggestions for articles. Each Journal issue contains one article each on PostScript and Acrobat. What topics would you like me to write about?

Questions and Answers. Do you have any questions about Acrobat, PDF, or PostScript? Feel free to email me about. I'll answer your question if I can. (If enough people ask the same question, I can turn it into a Journal article.)

Please send any comments, questions, or problems to:

john@acumentraining.com