Table of Contents

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 4.

Journal feedback: suggestions for articles, questions, etc.

Compressed PostScript

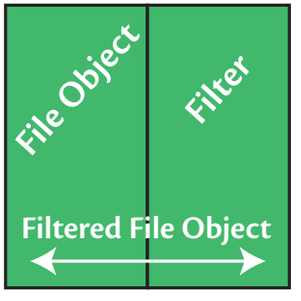
One of the most immediately obvious benefits of PDF compared to other file formats is that it is very compact; a document expressed as a PDF file is usually smaller—and often much smaller—than the same document embodied in other formats. There is a simple reason for this: nearly all of the contents of a PDF file are Flate, LZW, or otherwise compressed.

It turns out that you can easily do the same thing in PostScript, using the filter mechanism. The process is quite simple and is one that I recommend in circumstances that involve very large PostScript streams, such as variable data printing.

PostScript Filters: A Review

As you recall from your PostScript classes, a PostScript filter is a black box through which data can be read or written; data passing through the filter is modified in some manner specific to that filter: compressed, uncompressed, or converted between binary and some flavor of ASCII.

Typically, you attach a filter to a file object, yielding something I call a *filtered file object;* a filtered file object looks to PostScript exactly like a regular file object and may be read from or written to using the standard PostScript file operators. However, any data you read from or write to this object passes through the filter and is uncompressed (or whatever).



Applying a Filter

You attach a filter to a file object with the *filter* operator.

```
fileobj  /filtername  filter  =>  filteredFileobj
```

The *filtername* in the line above is one of several predefined names, listed at right. The *filter* operator attaches the named filter to the supplied file object and returns the new filtered file object.

The decode filters listed at right are all required by the PostScript specification. (PostScript also defines a corresponding series of encoding filters, but we shall ignore those in this article.) PostScript includes filters for many of the most commonly-used compression methods.

PostScript Decode Filters

Transmission filters (convert ASCII to binary)	
ASCIIHexDecode	ASCII85Decode
Uncompression filters	
RunLengthDecode	LZWDecode
FlateDecode	CCITTFaxDecode
DCTDecode	
Pass-through filters (data remains unchanged)	
SubFileDecode	ReusableStreamDecode

In driver output, you will most often see filters used in printing images. The image dictionary's *DataSource* entry is usually *currentfile* with a compression or transmission filter attached:

```
<<   /ImageType 1
     /Width     1200
     ...
     /DataSource currentfile /ASCII85Decode filter
>> image
```

Encode & Decode Filters

PostScript defines both *Encode* and *Decode* filters.

Encode filters convert raw data into a modified form; thus, the LZWEncode filter compresses the data that passes through it.

Decode filters—those listed in the table—convert encoded data back to its original form; the LZWDecode filter uncompresses the data passing though it.

JPEG

It isn't obvious from the name, but the DCTDecode filter implements JPEG uncompression.

Concatenating Filters

As I said earlier, a filtered file object is seen by PostScript as just another file object; this means that you can apply another filter to it. As an example: image data is very often both compressed (flate, say) and converted to ASCII (preferably ASCII85). The *image* operator would have to read such data through two filters: first the ASCII85Decode and then the LZWDecode filter:

```
<<   /ImageType 1
     /Width     1200
     ...
     /DataSource currentfile /ASCII85Decode filter /FlateDecode filter
>> image
```

The data read from *currentfile* would first be un-ASCII85'ed and then un-LZW'ed, yielding the original, raw image data.

Compressed PostScript

Mostly we use filters to uncompressed *data*. However, you can also use filters to execute compressed PostScript code. Just execute the contents of a file object (currentfile, for example) through an uncompression filter; the compressed PostScript code will be converted back to its original form and then handed to the interpreter.

The code is surprisingly simple:

```
currentfile                 % Get the current PS source
/LZWDecode filter           % Attach the LZWDecode filter
cvx                         % Convert the filtered file object to executable
exec                        % and execute it
... LZW-compressed PS code goes here ...
```

That's all there is to it.

You will need to have compressed the PostScript code ahead of time, of course. You may also find it necessary—because of network communication requirements, perhaps—to convert the compressed code to ASCII. You accommodate this simply by sandwiching a transmission filter between *currentfile* and the uncompression filter:

```
currentfile                 % Get the current PS source
/ASCII85Decode filter       % Attach the transmission filter
/LZWDecode filter           % Attach the uncompression filter
cvx                         % Convert the doubly-filtered file object to executable
exec                        % and execute it
... LZW-compressed PS code goes here ...
```

Now, the incoming stream will be passed through both filters before going to the interpreter.

For Example

The reason for doing this, of course, is to reduce the size of the PostScript stream making up our print job. This is particularly important for variable data printing, where print streams run into the many tens of gigabytes.

In the Variable Data PostScript class, we ultimately end up creating a PostScript stream that prints invoices for a fictional business. In outline, the output looks like this:

```
%!PS-Adobe-3.1
%%   ...
     ... Header comments
     ...
%%BeginProlog
     ...
     ... Definitions ...
     ...
%%EndProlog

%%Page: 1 1
<<
     ... Customer 1 data...
>> DoBill

%%Page: 2 2
<<
     ... Customer 2 data...
>> DoBill

...
...
```

A print run that generates 1,000 bills amounts to 292k of PostScript code. (Keep in mind an actual billing run for a major corporation consists of millions of individual bills.)

If we LZW compress the stream's Script (everything after the %%EndProlog), we now have this:

```
%!PS-Adobe-3.1
%%   ...
     ... Header comments
     ...
%%BeginProlog
     ...
     ... Definitions ...
     ...
%%EndProlog

currentfile /LZWDecode filter cvx exec
Ä    D•   ú :ƒh°fiS:7NÜQÃ¬vä ¶`h x
HDÚaÑÚo:ù<?>Y<§m ë‡≤©B6e7ÊyΠ€”
...
...
```

This version of the stream came to 86k, about 30% the size of the original.

Performance Considerations

Surprisingly, the execution of the LZW-compressed stream is no slower than that of the original, uncompressed PostScript. In fact, in all cases I have been able to test, the compressed stream executes somewhat *faster* than the original. I can only assume that the transmission time saved in reducing the size of the stream is greater than the time lost in uncompressing the stream prior to execution.

Use It When?

The circumstances under which it is useful to compress a PostScript stream are not that common. If you have very large streams or a great many smaller streams that must be archived in an executable state, then this is well worth doing.

The *Examine Document* Feature

I stumbled across this feature recently, surprised to find it was first introduced back in Acrobat 8.

Examine Document, which resides in the *Document* menu, scans through your document looking for a variety of hidden content within your PDF file. This includes metadata, artwork and text that extend (or completely reside) off the visible page, and text that is layered beneath and therefore hidden by other objects on the page.

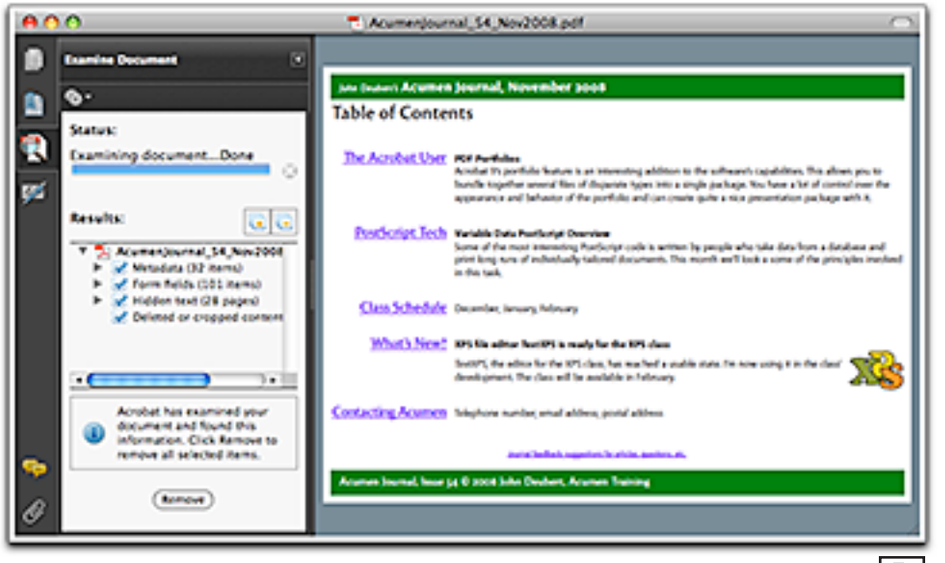
Having found these items, Acrobat give you the chance to remove them from the PDF file. The intent of the feature is security: removing sensitive information from a document that you didn't realize was there because it was hidden from view. However, I find removing this data can also sometimes significantly reduce the size of a PDF file without at all changing its appearance. And then there's the tidiness factor: it bothers me to have artwork and text taking up space in a PDF file to no purpose.

Let's see how to use it and what it does.

Using the Feature

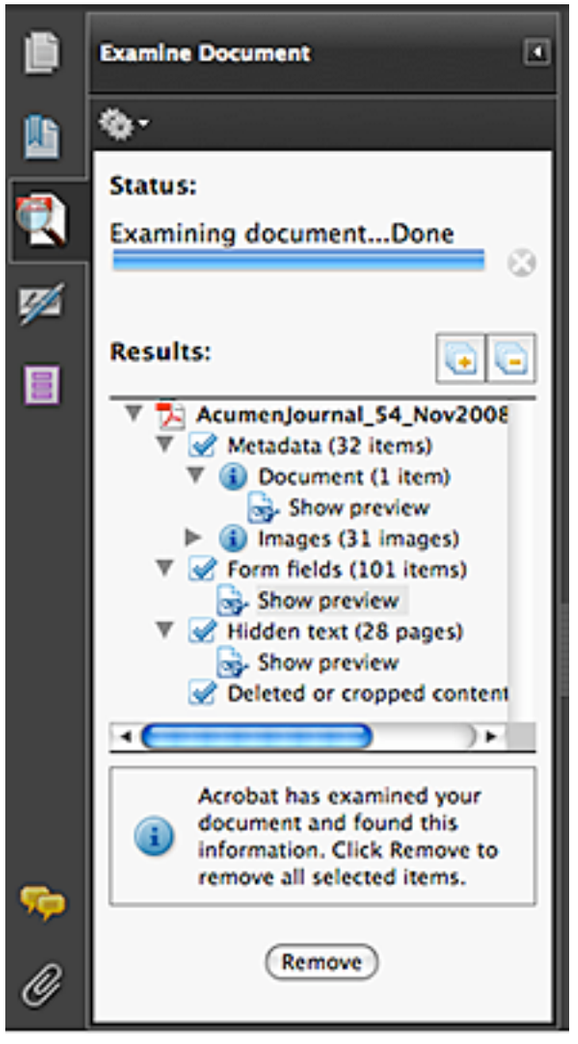
You invoke Examine Document by selecting, well, *Examine Document* from the View menu.

Acrobat immediately scans through the frontmost PDF file and then opens the *Examine* navigation pane, as at right.



This is a unremarkable pane, but useful. From top to bottom, it contains:

- A progress bar, indicating how the current activity (whatever that might be) is proceeding.
- A *Results* list of all the objects of interest found in the scan.
- A *Remove* button that will remove all or some of the found items from the PDF file.



The Results List

The Results list itemizes all the objects found during the scan. These objects are not visible on the page and so may be removed without changing the appearance of the PDF document.

This is a hierarchical list, organized into categories. The set of categories that appear will depend upon the results of the scan. Categories I routinely see include:

- **Metadata** - Itemizes metadata residing in the file, including such things as the document's title and author.
- **Form Fields** - This lists the form fields that appear in the document. If you choose to remove these, they will be replaced with PDF graphic objects that look the same on the page, but are no longer active form fields.
- **Hidden Text** - This is text that is overlain by other graphics and so is invisible on the page.
- **Deleted or cropped contents** - This includes graphic items that lie wholly or partially off the page and items that have been cropped so they are only partially visible. It also includes items that have been removed using the Acrobat Touch Up tool (or other equivalent utility) but that still remain in the PDF file. (Remember that an item deleted from a PDF document may still physically reside in the file.

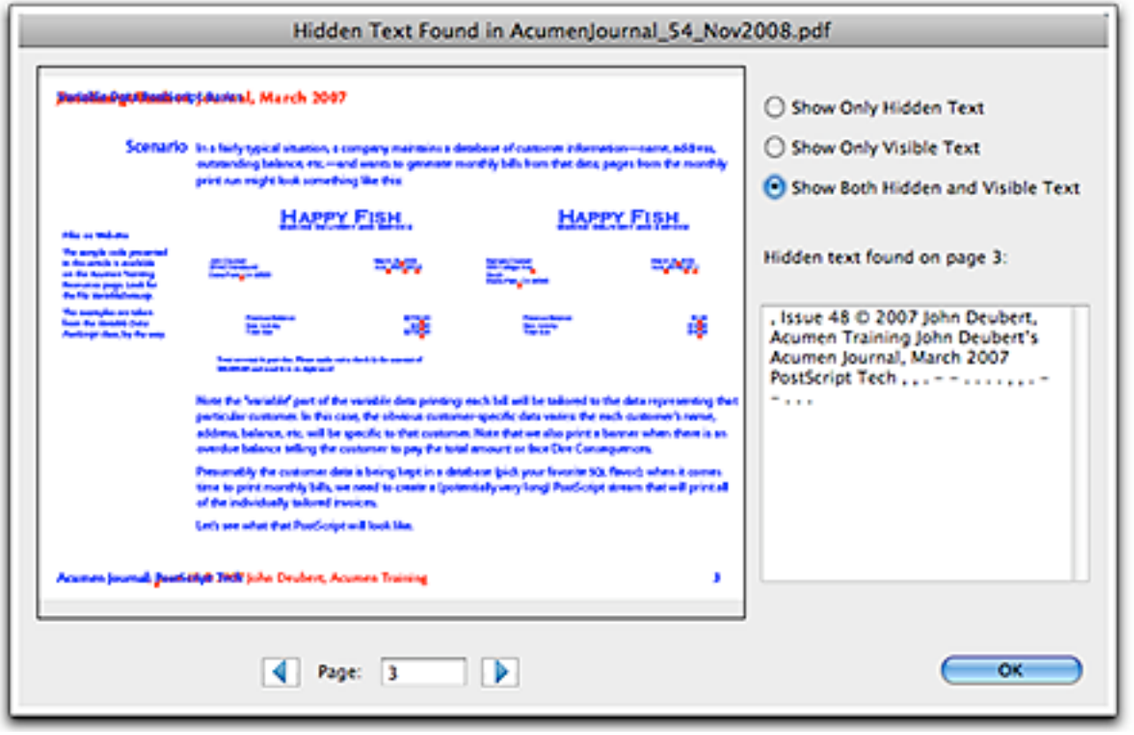
- **Comments and Markups** - This lists any sticky notes or other annotations attached to the document. Although these items are not actually "hidden," Acrobat does not consider them to be part of the document and so are candidates for removal.

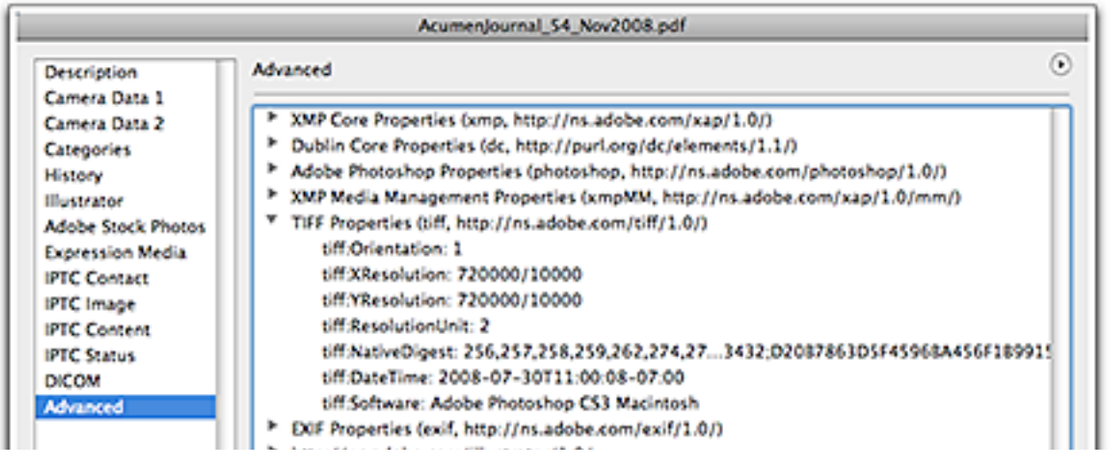
Examining Results

If you click on one of the items in the Results list, Acrobat displays a dialog box that shows you the details of the hidden item.

At right, for example, is the dialog box that shows you the hidden text the examination found.

One surprise is if you click on one of the images in the Metadata category. You get a dialog box showing you the metadata associated with the image (lower right), not the image itself. (The image *is* highlighted in the document window, however, so you can tell which image it is.)



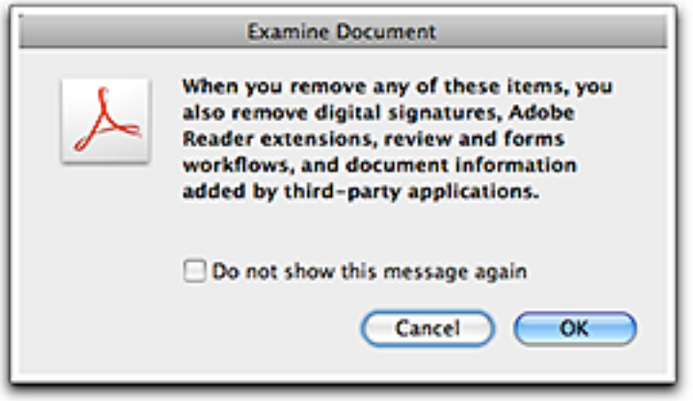


Removing the Objects

The *Remove* button at the bottom of the Examine pane will do just what it says: remove the items you have selected in the Results list (or remove them all, if you haven't selected any items).

When you click the Remove button, Acrobat confirms that you want to do this, warning you of some of the consequences of the action.

Click OK and Acrobat will run through the file, removing the items you specified.



Appropriate Use

Removing invisible items from a PDF file can significantly reduce the size of that file. It also ensures that you are not distributing a PDF file that has unexpected information in it, such as items deleted from earlier drafts or commentary that shouldn't be distributed as part of the final document.

Schedule of Classes, February – April 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	Feb 16–19		Apr 6–9
PDF 2: Advanced File Content			
Support Engineers' PDF	Feb 2–3	Mar 12–13	

PostScript Classes

PostScript Foundations	Feb 9–13	Mar 23–27	
Advanced PostScript		Mar 30–Apr 2	
Variable Data PostScript			
Troubleshooting PostScript		Mar 9–11	

XPS Classes

XPS File Content and Structure			May 4–7

Course Fee Classes cost $2,000 per student, except for *Troubleshooting PostScript* ($1,500) and *Support Engineers' PDF* ($1,000). There is a 10% discount for signing up three or more students. 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?

First XPS Class May 4

The first *XPS File Structure and Contents* class is now scheduled for May 4 at Acumen Training's classroom site in Costa Mesa, California (near the Santa Ana/John Wayne airport). The four day class will present in-depth coverage of the essentials of the XML-based Paper Specification.

An day-by-day course outline is still a few weeks away (keep an eye on the website), but the broad topics will include:

- 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* will concentrate 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.

I'm very excited about this class and 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 inexplicably make you think of that lemmings have the right idea?

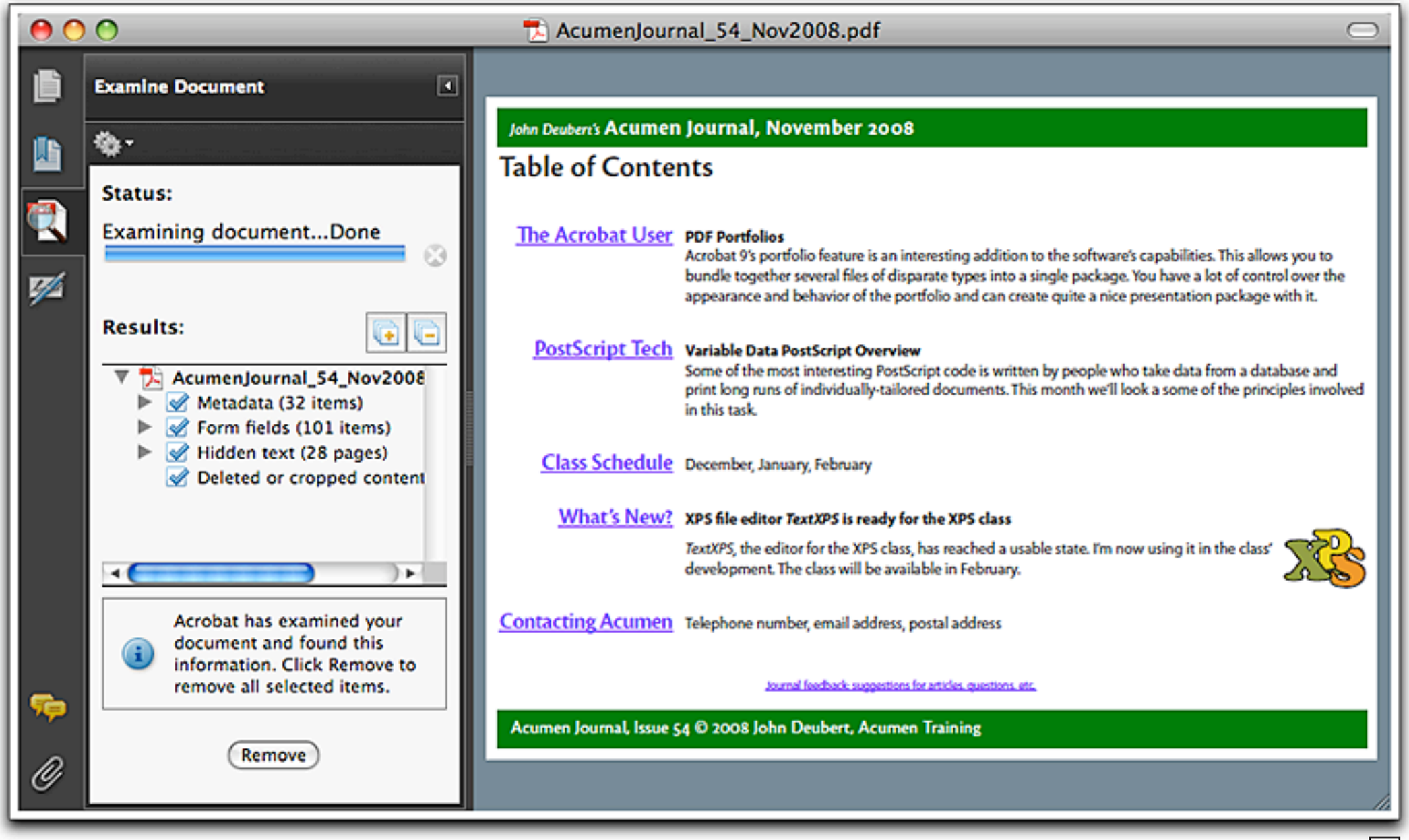
Suggestions for articles. Each Journal issue contains one article each on PostScript and Acrobat. What topics would you like me to write about?

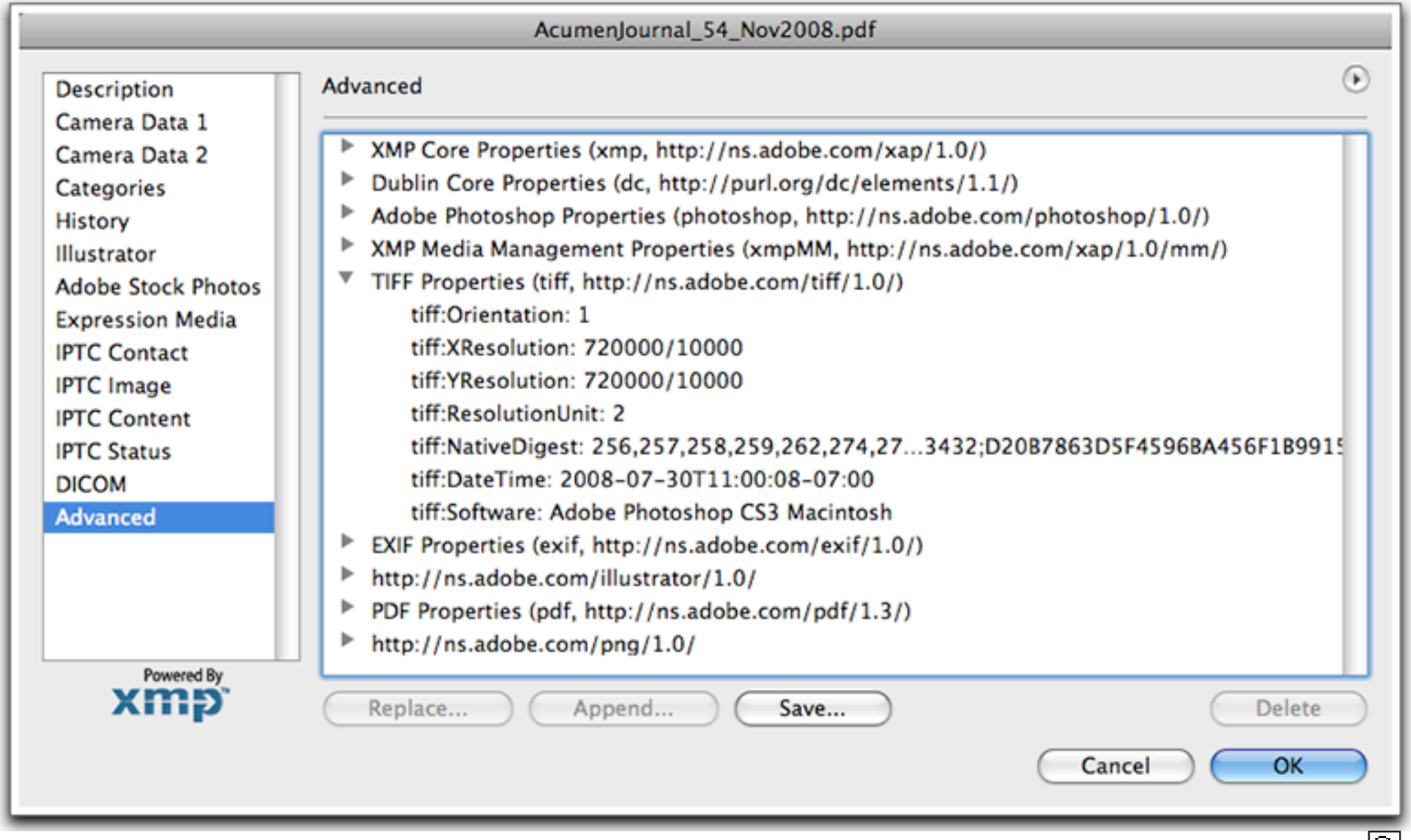
Questions and Answers. Do you have any questions about Acrobat, PDF, or PostScript? Feel free to email me about. I'll answer your question if I can. (If enough people ask the same question, I can turn it into a Journal article.)

Please send any comments, questions, or problems to:

john@acumentraining.com

Examine Document Pane




AcumenJournal_54_Nov2008.pdf
Description
Camera Data 1
Camera Data 2
Categories
History
Illustrator
Adobe Stock Photos
Expression Media
IPTC Contact
IPTC Image
IPTC Content
IPTC Status
DICOM
Advanced
Powered By
xmp
Advanced
XMP Core Properties (xmp, http://ns.adobe.com/xap/1.0/)
Dublin Core Properties (dc, http://purl.org/dc/elements/1.1/)
Adobe Photoshop Properties (photoshop, http://ns.adobe.com/photoshop/1.0/)
XMP Media Management Properties (xmpMM, http://ns.adobe.com/xap/1.0/mm/)
TIFF Properties (tiff, http://ns.adobe.com/tiff/1.0/)
tiff:Orientation: 1
tiff:XResolution: 720000/10000
tiff:YResolution: 720000/10000
tiff:ResolutionUnit: 2
tiff:NativeDigest: 256,257,258,259,262,274,27...3432;D20B7863D5F4596BA456F1B9915
tiff:DateTime: 2008-07-30T11:00:08-07:00
tiff:Software: Adobe Photoshop CS3 Macintosh
EXIF Properties (exif, http://ns.adobe.com/exif/1.0/)
http://ns.adobe.com/illustrator/1.0/
PDF Properties (pdf, http://ns.adobe.com/pdf/1.3/)
http://ns.adobe.com/png/1.0/
Replace...
Append...
Save...
Delete
Cancel
OK