Table of Contents

pdfmark and Transparency

Sample Files

As usual, the PostScript code for this month's article is available on the Acumen Training Resources page. Look for the file *Transparent.ps.*

PostScript has always implemented a completely opaque imaging model; when you paint something on the page, you completely obscure anything that was on the page already.

That is, unless you are writing PostScript specifically for sending to Acrobat Distiller, which will convert the PostScript to a PDF document; in that case, you *can* write PostScript that paints onto the page transparently (Figure 1) using a Distiller-only operator called *pdfmark*.

Let's look.

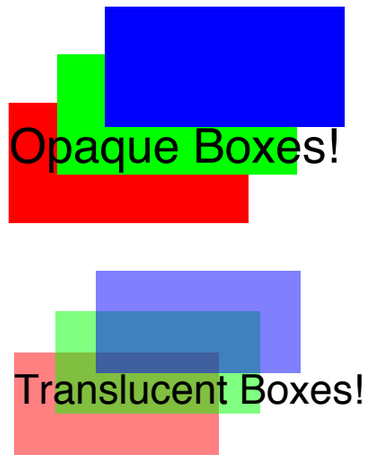


Figure 1. The Distiller-only *pdfmark* operator lets you specify transparency in your PostScript code.

pdfmark

The *pdfmark* operator lets you insert PDF-specific commands into a PostScript program; the operator only exists in the Distiller version of PostScript (and some other PostScript to PDF conversion tools, such as *PDF Creator)*. The operator looks like this:

```
[  key0 value0 key1 value1 ... /keyword  pdfmark
```

The arguments start with a mark object, which may be placed there with an open bracket, as above, or with a call to the PostScript *mark* operator. The mark is followed by a variable number of key-value pairs, followed by a keyword that indicates what *pdfmark* is supposed to do. Table 1 presents a sample of just a few of the keywords *pdfmark* understands.

pdfmark Reference

There are a *lot* of keywords you can use with *pdfmark;* they're worth exploring.

The complete capabilities of *pdfmark* are documented in the *pdfmark Reference Manual,*available on Adobe's website. Here's the download link.

Table 1. Some *pdfmark* Keywords

Keyword	*Meaning*
ANN	Add an annotation to a specified page
OUT	Add a PDF bookmark to a specified page. (Short for "outline.")
ARTICLE	Add an article box to the page
DOCINFO	Specify the document info for the PDF output.
SetTransparency	Specify transparency (subject to *gsave* and *grestore).*

So, for example, the PostScript code below creates a PDF file with an annotation on the page (Figure 2).

```
[
   /Rect [ 400 500 500 600 ]
   /Subtype /Text
   /Title (Sample Annotation)
   /Contents (This text annotation was created with a pdfmark.\n\nCool, huh?)
   /ANN
pdfmark

1 0 0 setrgbcolor
100 500 200 100 rectfill
0 1 0 setrgbcolor
140 540 200 100 rectfill
0 0 1 setrgbcolor
180 580 200 100 rectfill
```

The key-value pairs supplied to *pdfmark* in support of the ANN keyword are probably pretty obvious: Rect defines where on the page the annotation icon should go, Subtype indicates the type of annotation it is, etc.

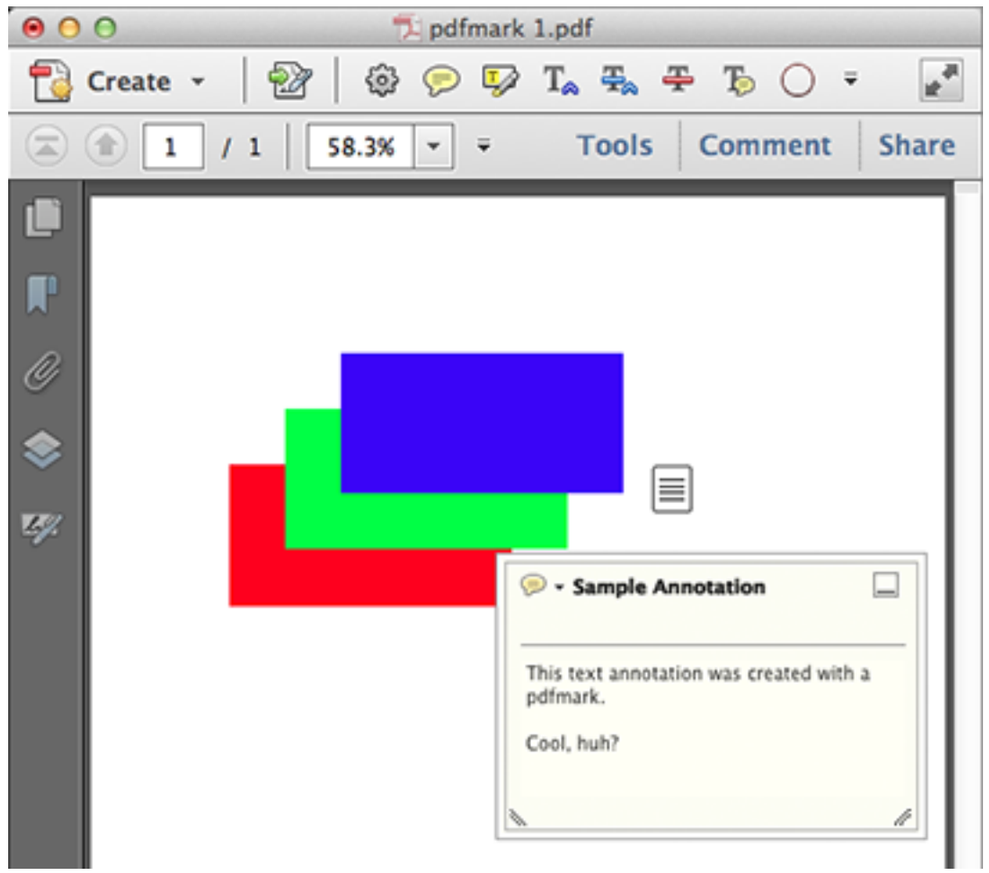


Figure 2. The sample call to*pdfmark* created a text annotation on the PDF output page.

Transparency

So, what about transparency, then?

The SetTransparency keyword tells Distiller to set the opacity used by the stroke and fill operators. Opacity is a value between 0 (completely transparent) and 1 (compeletely opaque.) Interestingly, there are separate opacity values for stroking and filling, which matches how PDF does things.

PDF 1 Students…

If you've taken the PDF 1 class, you probably recognize */ca* and */CA* as the keys used to specify opacity in a PDF ExtGState object.

In fact, *all* of the PDF opacity-related key-value pairs (/SMask, /BM, etc.) may be used with the SetTransparency pdfmark.

The relevant key-value pairs for *pdfmark* are:

/ca	float	The opacity that should be used for filling shapes.
/CA	float	The opacity that should be used for stroking.

Thus, the following code:

```
[
    /ca .25
    /CA .75
    /SetTransparency
pdfmark
```

sets the fill opacity to 0.25 and the stroke opacity to 0.75.

Here's an example using this call to *pdfmark:*

```
300 400 translate
gsave
5 setlinewidth

0 .125 .9
{  1 1 sethsbcolor          % Draw some radial lines
   -200 0 moveto 200 0 lineto
   stroke
   22.5 rotate
} for
```

```
[  /ca .25          % Set the fill opacity to 25%
   /CA .75          % Set the stroke opacity to 75%
   /SetTransparency
pdfmark

0 setgray           % Draw a (translucent) black circle
20 setlinewidth
0 0 150 0 360 arc
gsave
fill
grestore
stroke
```

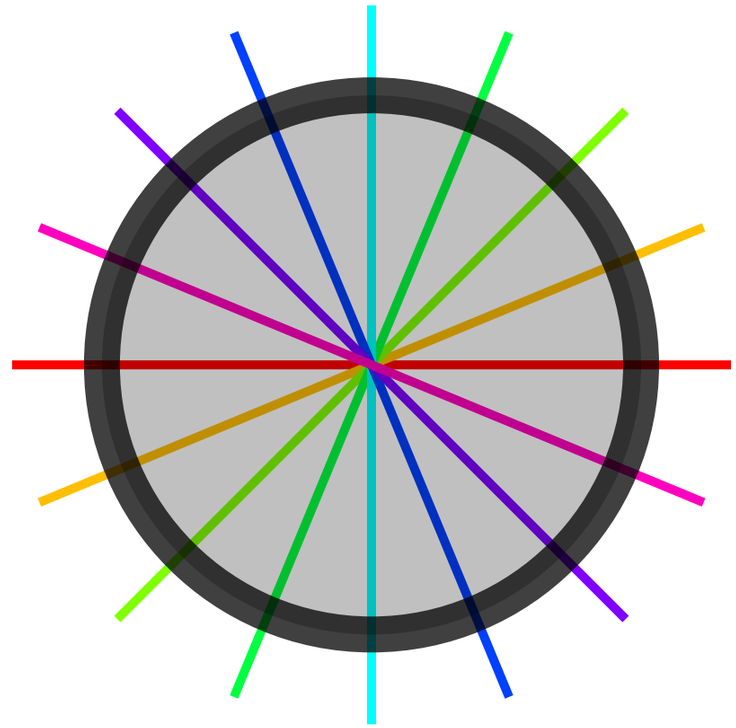


Figure 3. Here we used *pdfmark* to draw a translucent black circle on the page. Note that *stroke* and *fill* have separate opacity values.

The resulting PDF page looks like Figure 3. Note that the circle's stroked border has a higher opacity (0.75, to be precise) than its filled interior.

Life Should Be So Easy

Unfortunately, all is not skittles and beer; there is something you'll need to do before you can use the SetTransparency pdfmark.

By default, Distiller has transparency turned off; the call to *pdfmark* returns an *undefined* error, indicating that Distiller doesn't recognize the SetTransparency keyword. This isn't unreasonable; PostScript is intended primarily as a printing language and using transparency in PDF files intended for high-end printing is a frequent cause of failed print jobs.

Before you can use SetTransparency, you need to tell Distiller that it's okay. In principle, we should be able to do this with a call to *setdistillerparams:*

```
<< /AllowTransparency true >> setdistillerparams
```

However (and I don't know why this is so), this call to *setdistillerparams* doesn't seem to work; it has no effect on Distiller's willingness to recognize the /SetTransparency pdfmark; you still get an *undefined* error when you try it. (Just to head off some suggestions: I've already tried setting the compatibility to 1.6 and setting /LockDistillerParams to false.)

Happily, if weirdly, you *can* get SetTransparency to work by modifying the AllowTransparency entry in the Distiller settings file. Let's see how to do that.

Distiller Job Options

Distiller's main window has a prominent pop-up menu from which you may choose among previously-saved collections of settings (Figure 4). Each collection of settings is stored in a file on your hard disk; on the Mac, settings files are stored in ~/Library/Application Support/Adobe/Adobe PDF/Settings; in Windows, they're in the same folder as the Distiller executable (acrodist.exe) in a folder named "Settings" (Figure 5).

If you open one of these files in a text editor, you'll see that it contains PostScript code; specifically, it contains a call to *setdistillerparams* that, among other things, sets /AllowTransparency to false:

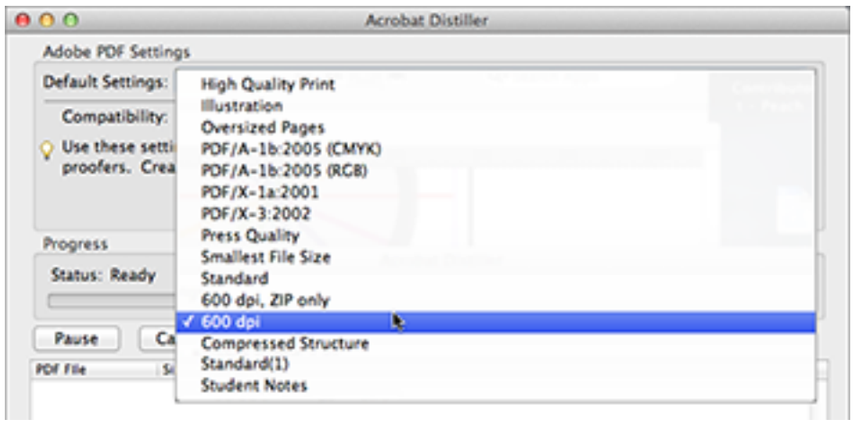
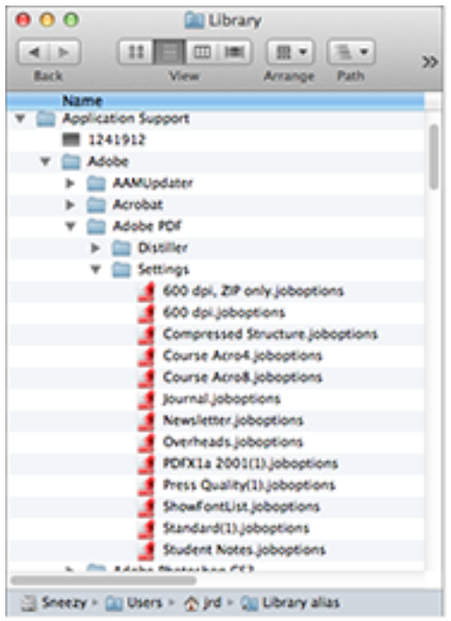


Figure 4. Distiller's main window has a pop-up menu of settings you can select. Each of these has a corresponding settings file on your disk.



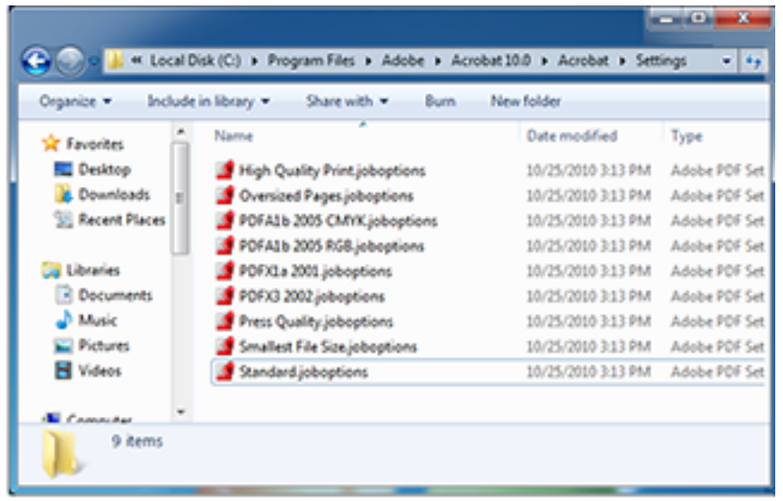


Figure 5. A folder full of Distiller settings files is stored among the Application Support folders on the Mac and in the Distiller executable folder in Windows.

```
<<
   /ASCII85EncodePages false
   /AllowTransparency false
   /AutoPositionEPSFiles false
   ...
   ...
>> setdistillerparams
```

To allow the use of the SetTransparency pdfmark, just set the value of /AllowTransparency to true within the settings file. Then, when you distill your PostScript code, select the settings whose file you modified.

Everything works.

It's a Mystery

The mystery here is *why* this works, when your own call to *setdistillerparams* will not. The settings file executes before your own code, so your call to *setdistillerparams* should override that in the settings file. In fact, it *does* do so; checking the value of *AllowTransparency* after the call to *setdistillerparams* indicates that its value is successfully being set to true.

```
<< /CompatibilityLevel 1.6
   /AllowTransparency true
>> setdistillerparams
currentdistillerparams
dup /CompatibilityLevel get ==     % <== CompatibilityLevel 1.6
/AllowTransparency get ==          % <== AllowTransparency true
```

And yet, the SetTransparency *pdfmark* call still doesn't work.

Go figure.

Still, setting AllowTransparency in the settings file *does* work and specifying transparency in my PostScript code is occasionally very useful (and strangely fun).

If anyone has a notion as to why in-line calls to *setdistillerparams* don't work, I'd love to hear it.

Automating Acrobat X with Actions

You might also like "Diving Into Acrobat"

If you find the *Acumen Journal* useful, you might like my on-line series, *Diving Into Acrobat,* on Peachpit Press' InFormit website. Recent articles include:

- Fixing Files with Preflight
- Two Exotic Annotations
- A Color Management Primer

You can read the articles here.

It will probably not astonish anyone that I use Acrobat a *lot*. Most of the documents I produce—student notes, lecture overheads, invoices, expense reports, almost everything, really—end up as PDF files for storage and distribution.

There are certain tasks that frequently recur in my Acrobat-related life: setting Open Options, making searchable indices, etc. Since these repeated activities are often almost the same from one time to the next, I generally look to see if I can automate these processes using Acrobat's batch processing mechanism.

Acrobat has had a batch processing ability for a very long time, now. Usually, each new version of Acrobat implemented back processsing differently than its predecessor and Acrobat X is no different. Adobe has significantly changed the mechanism by which you create and use batch processes, which Acrobat now calls "Actions."

Actually, I think that with Acrobat X, Adobe has done its best job ever of making Actions easy to create and use without sacrificing access to the more exotic abilities. Let's look it over.

The Action Wizard

Why "Wizard?"

Properly speaking, the Actions Wizard panel isn't a "wizard" in the customary sense of the word.

A wizard is usually a series of dialog boxes presented in sequence to step the user through a complex process. The Acrobat X Actions Wizard panel is just a set of tools with no particular order implied.

So, why "Actions Wizard?" I presume someone just thought it sounded good.

As with so much else in Acrobat X, the Actions controls have been moved from the toolbar to the Tools pane; the Action Wizard panel (Figure 1) contains clickable buttons for all the defined actions, as well as a pair of tools that let you create a new action and edit the existing ones.

If you click on any of the actions listed below the "Actions" separator (such as the "Prepare for Review" action visible in Figure 1), Acrobat immediately executes that Action.

What we're interested in this month, however, is the "Create New Action" control at the top of the panel (still Figure 1). This is what you use to create a new Action.

In this article, we'll create an Action named "Journal Open Options" that I can use to automatedly set the Acrobat Open Options for an *Acumen Journal* PDF file, such as the one you're reading right now.

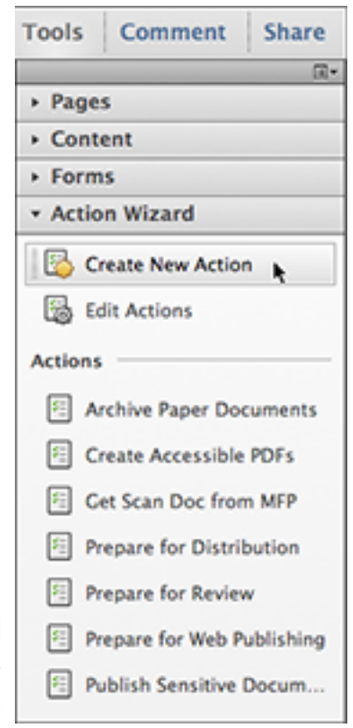


Figure 1. The Action Wizard panel contains all the currently-defined Actions

Creating a New Action

1. Click Create New Action

Start by clicking on the Create New Action tool in the Actions Wizard panel. Acrobat will display the Create New Action dialog box (Figure 2).

2. Choose a "Target"

First, we tell Acrobat on what file it should be operating by selecting an item in the "Start with" pop-up menu (Figure 3, next page).

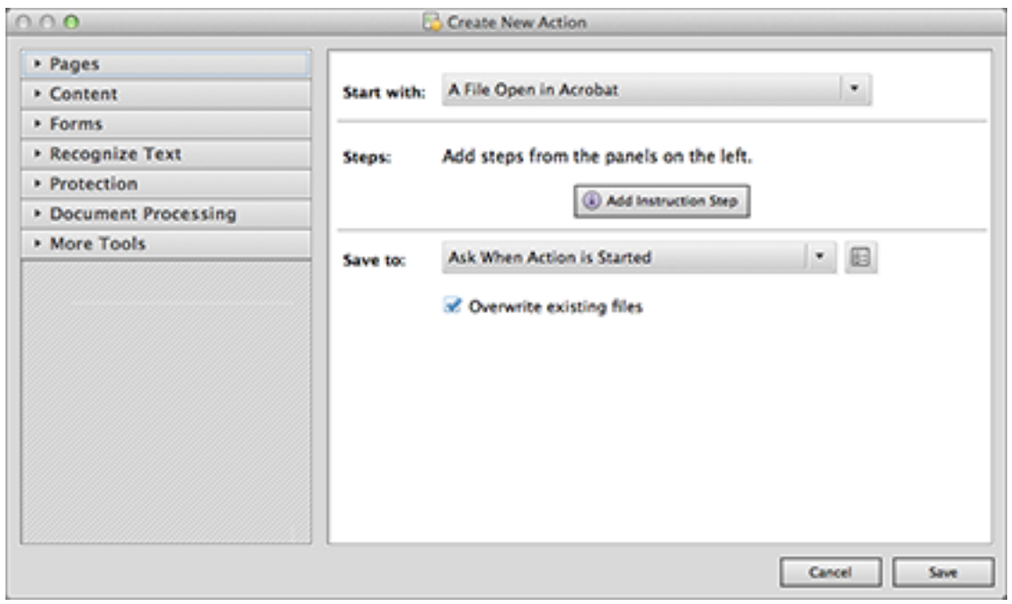


Figure 2. The Create New Action dialog box provides the controls with which you create a new Action. That explains the name, anyway..

Choose a target that fits the Action you are creating. (Note that I'm presenting these slightly out of order.)

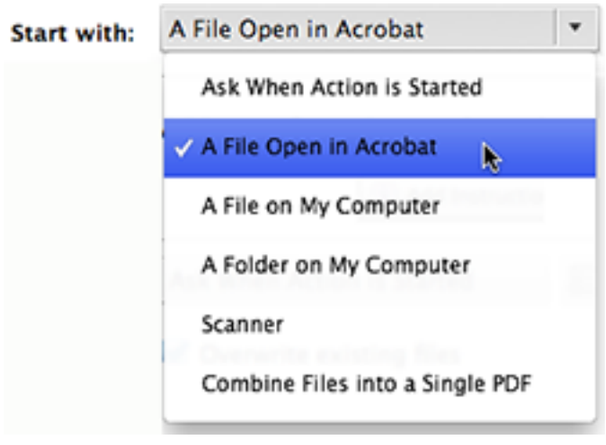


Figure 3. The *Start with* pop-up menu specifies the Action's target.

- *A File Open in Acrobat* is the most common choice. This says that the Action should be applied to the frontmost open document.
- *A File on My Computer* tells Acrobat to present you with a standard "Pick a File" dialog box and let you select a file that the Action should act upon.
- *A Folder on My Computer* tells Acrobat to present you with a standard "Pick a Folder" dialog box and let you select a folder. The Action will act upon all PDF files within that folder.
- *Scanner* tells Acrobat to scan a document to a PDF file (using a scanner available to your computer) and then apply the Action to the result.
- *Combine Files into a Single PDF* presents you with Acrobat's standard Pick a Series of Files dialog box. The files you choose are combined into a single PDF file and the Action is applied to the result.
- *Ask When Action is Started,* as you might imagine, asks at runtime which of the preceding targets you want to use in the Action.

For myself, I don't think I've ever chosen anything other than "A File-" or "A Folder on my Computer."

3. Choose Your Actions

Having told Acrobat what file the Action should apply to, now you need to tell it what the Action should do to that target. Click on the Add Instruction Step button and select an activity from the set of panels occupying the left side of the Create Action dialog box (Figure 4). In my case, I'll pick Set Open Options in the Document Processing panel, as in the figure.

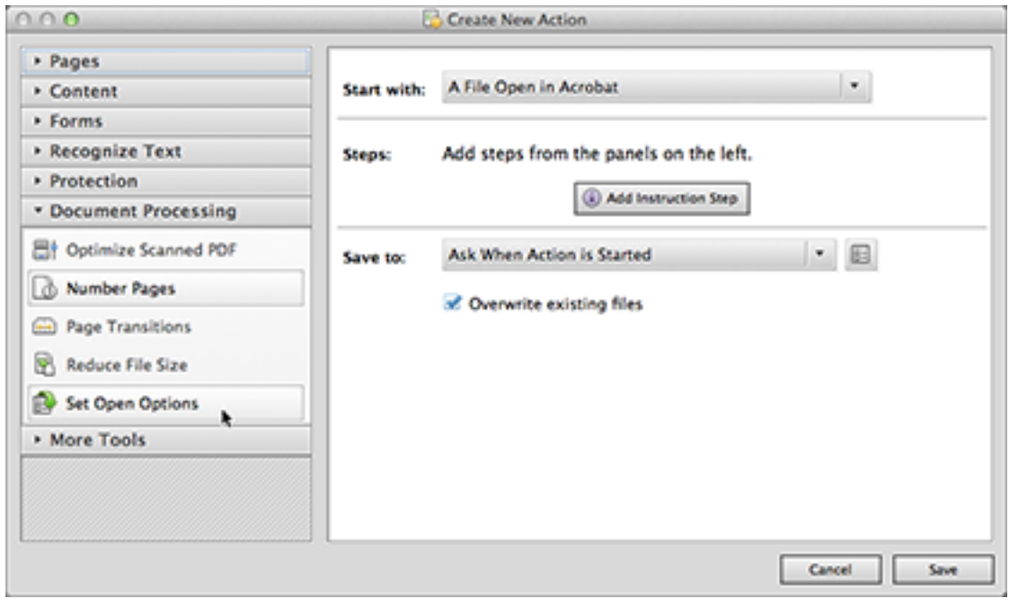


Figure 4. The pane making up the left side of the dialog box contains all the activities you can add to an Action.

Acrobat will add your selection as a new step in the Action you are creating (Figure 5). You can add as many steps as you wish to your Action.

The set of available activities you can add to an Action is quite large; I'll leave you to explore them.

4. Choose Options

In most cases, each step you add to your actions will have an Options icon associated with it, visible in Figure 5. If you click on this icon, Acrobat will present you with a dialog box specific to that step, allowing you to specify some set of options that affect how the step works; in the case of my Document Open Options step, the dialog box is almost the same as the Open Options panel of the standard Document Settings dialog box (Figure 6, next page).

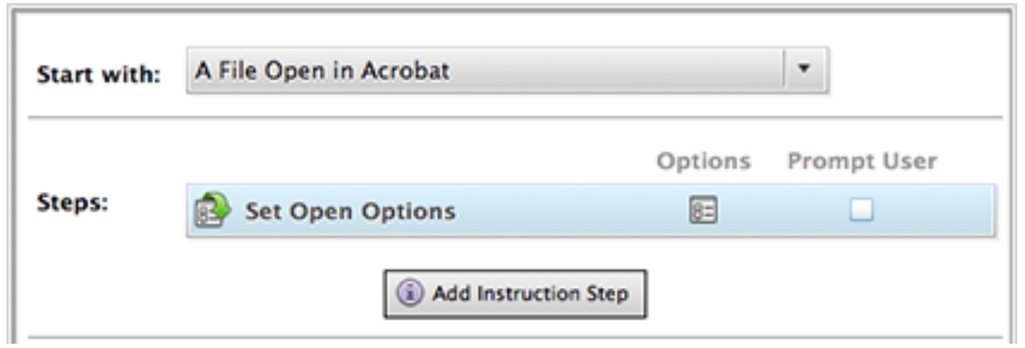


Figure 5. The Create New Action dialog box lists the steps currently added to your Action. Most steps have an Options icon that lets you specify details for that step.

I've selected Open Options that will make the PDF file work well as an Acumen Journal document: center the window on the screen, resize it to fit the document page, etc. These settings will be applied when I run my Action.

Close this dialog box, as usual, by clicking the OK button.

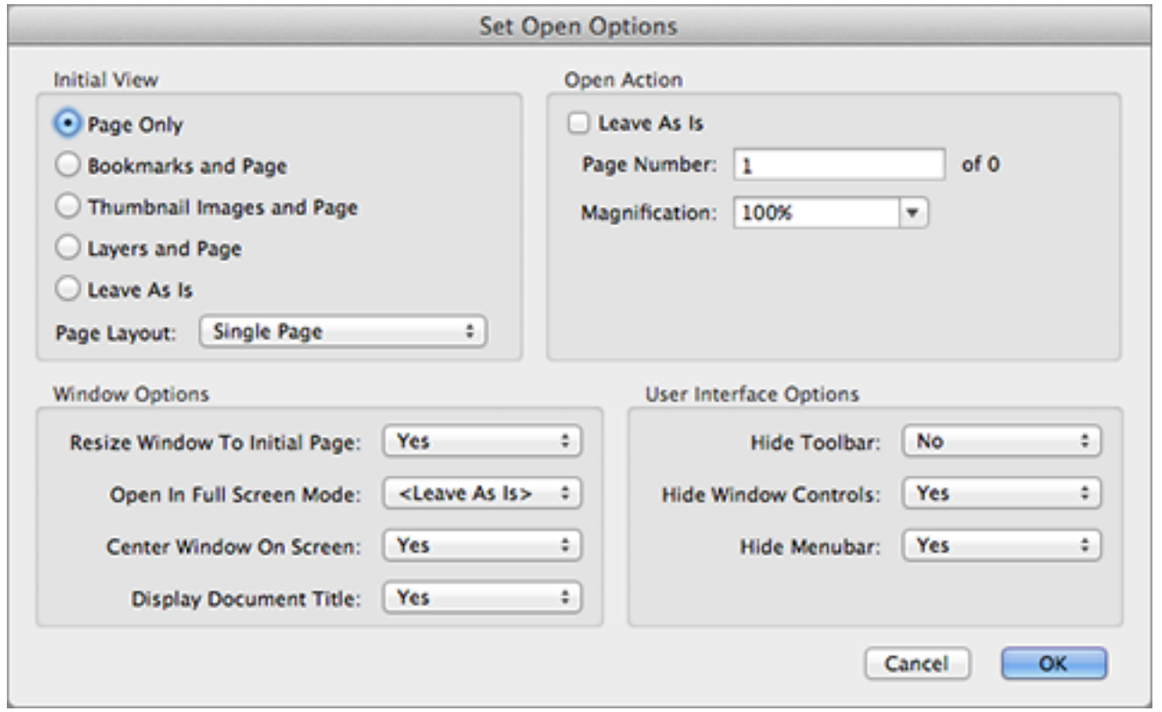


Figure 6. When you click on the Options icon associated with a step in your Action, Acrobat presents you with a dialog box with settings specific to that step.

5. What Shall We Do with the Modified File?

When the action has finished executing, what should happen to the resulting, modified PDF file? The possibilities are listed in the *Save to* pop-up menu (Figure 7). These are all pretty straightforward, so I'll let you experiment on your own.

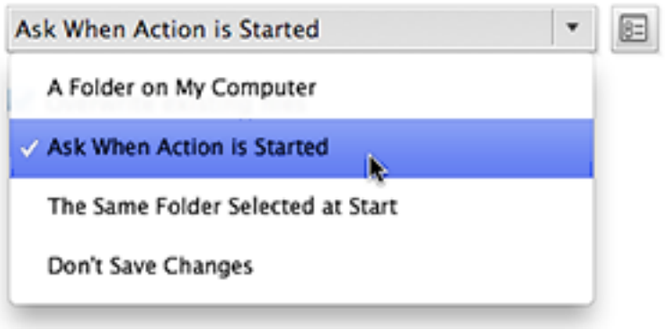


Figure 7. The *Save to* pop-up menu tells Acrobat what to do with the PDF file when the Action is finished.

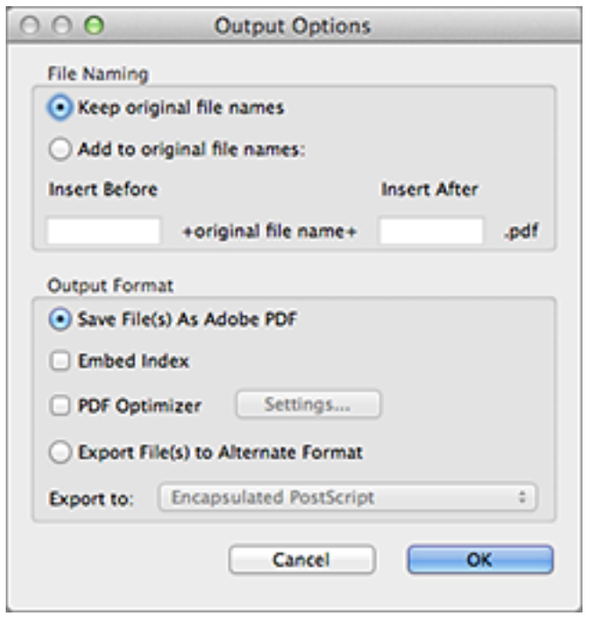


Figure 8. The Output Options specify the details of how the resulting, modified PDF file should be saved.

I shall, however, call your attention to the small icon just to the right of the pop-up menu in Figure 7. If you click on this icon, Acrobat presents you with a dialog box that lets you specify the details of how the new PDF file should be saved (Figure 8). Again, most of these controls

are easy to understand and you've probably seen their equivalent in many other applications. Two controls are worth mentioning in more detail, however:

- The *Embed Index* checkbox tells Acrobat to embed in the file an index of all the words the file contains. This makes it vastly faster to search for text in that file and is worth doing if this is a document that will often be searched (a catalog, perhaps). Be a little wary of this option, however; it can substantially increase the size of the PDF file.

- The *PDF Optimizer* checkbox and its attendant Options button lets you apply a variety of optimizations to the file, most of whose net effect is to reduce file size, sometimes by quite a lot.

6. Save the Action

Finally, returning to the Create New Action dialog box, we click on the Save button to save our Action. Acrobat presents us with the Save Action dialog box (Figure 9), which lets us specify a name and description for the new Action. Type in something appropriate, click OK, and you're done.

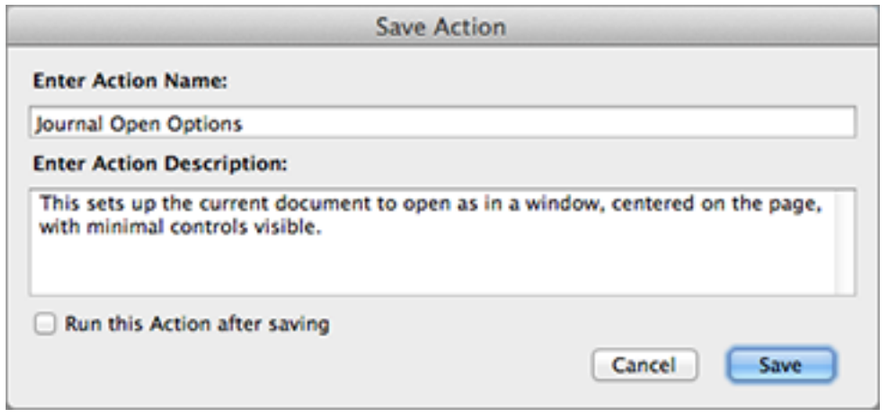


Figure 9. Finally, the Save Action dialog box lets you supply a name and description for your newly-minted Action.

7. Use the Action

The new Action now appears in the Actions Wizard pane, along with all the other Actions (Figure 10). Just click on it and Acrobat will carry out the Action.

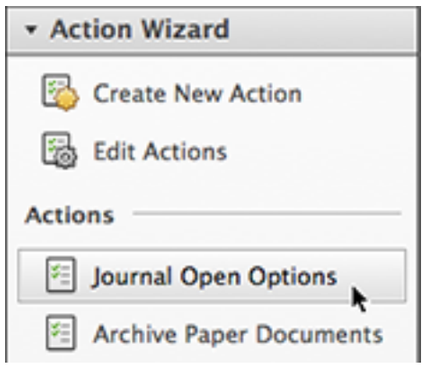


Figure 10. The new action appears in the Action Wizard panel. Just click on the button to execute the Action.

Nice Job

Adobe did a good job with the Actions implementation in Acrobat X. Designing a user interface for an application's macro facility is surprisingly tricky. Coming up with a set of controls that give technical novices access to all the possibilities of the Actions mechanism is a fine exercise in layout, design, and applied psychology.

I think they were more successful at it than in any earlier version of Acrobat.

Schedule of Classes, January 2012– March 2012

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:

- *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.

PDF Classes

PDF 1: File Content and Structure	Jan 17-20	Feb 28-Mar 2	
PDF 2: Advanced File Content			
Support Engineers' PDF		Feb 2-3	

PostScript Classes

PostScript Foundations	Jan 9-13	Aug 8–12	Mar 5-9
Advanced PostScript			Mar 12-15
Variable Data PostScript		Feb 13-17	
Troubleshooting PostScript	Jan 30-Feb 1		

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?

The JavaScript Book on Sale in January

Learning Acrobat JavaScript, an update to the old *Extending Acrobat Forms with JavaScript* book, will be available for sale in January. It will be available on the Acumen Training website and Amazon.com (although the latter isn't in place yet).

There will also be a two-chapter sample available free for the downloading in December.

If you want to be notified when the book's on sale, drop me an email at john@acumentraining.com; put "Javascript" somewhere in the subject.