Table of Contents

Acumen Training

The Acrobat User

Fixing Acrobat 5's Line Widths
Acrobat 5 tends to draw lines too thick, compared to Acrobat 4. This month we'll see how to fix this.

PostScript Tech

Changing Character Widths with a *Metrics* Dictionary
PostScript provides a little-used mechanism for overriding the character widths in a font. By providing a dictionary named *Metrics,* you can specify your own character widths.

Class Schedule

September-October-November-December
Where and when are we teaching our Acrobat and PostScript classes? See here!

What's New?

Drop by at Seybold.
I'll be teaching three classes at the San Francisco Seybold Seminars in September. Come by and say Hello!

Contacting Acumen

Telephone number, email address, postal address, all the ways of getting to Acumen.

Journal feedback: suggestions for articles, questions, etc.

Fixing Acrobat 5's Thin Lines

I like Acrobat 5. It added a lot of useful features; its JavaScript interface is far richer than 4's; without a doubt, it is a significant improvement over Acrobat 4.

Except for one thing: when displaying a PDF document, Acrobat 5 consistently displays thin horizontal and vertical lines too thick, compared to those very same hairlines (in the very same PDF file) in Acrobat 4. They print fine; they just look bad on the screen.

For example, at right are a pair of screen shots, one each from Acrobat 4 and 5, taken from the March 2001 issue of the Acumen Journal. The Acrobat 5 underline strokes are far thicker than they should be.

Hairline in Acrobat 5

Same Hairline in Acrobat 4

Happily, this is fixable. There is a PostScript snippet that you can paste into your *prologue.ps* file that will make hairlines come out correctly when you make a PDF file. You can also add the same fix to an existing PDF file using an Action List in Enfocus' PitStop plug-in.

We'll discuss both of these methods in this month's *Acrobat User.*

Next Page ->

Acumen Journal: The Acrobat User

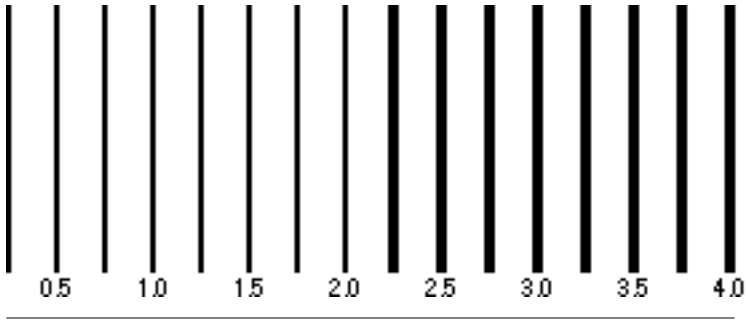
Background

PDF line widths are floating point numbers; that is, line widths can have a fractional values, such as 0.172 points. However, the computer screen on which the document is displayed creates its image out of discrete pixels, which may be only on or off and nothing in between.

When displaying a PDF file, therefore, Acrobat must decide how to paint fractional-pixel-wide lines on a screen; it must somehow round off the requested line width to an exact number of screen pixels.

There are several ways of doing this. Acrobat 5 seems to use a very conservative method that always turns on an even number of screen pixels for its line widths.

You can see this in the screen shot at right, taken of a PDF file in Acrobat 5; here we have a set of vertical lines, each 1/4 point thicker than the one before.



As you can see, even very thin lines are painted two pixels thick; what should be quarter- or half-point lines are being painted two points thick. (For comparison, there is a 1-pixel thick line running beneath the screen shot.)

At a specified line width of 2 1/4 points, Acrobat 5 jumps to painting the lines 4 pixels (and, therefore, 4 points) thick.

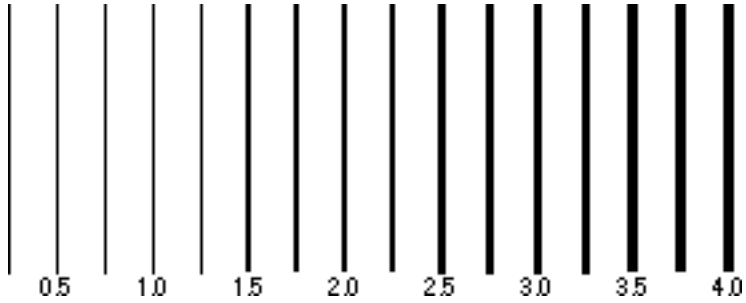
This is not a very accurate rendering of the desired line width.

Next Page ->

Automatic Stroke Adjustment

PostScript printers also have this problem, although the line width inaccuracies are less significant, because printer pixels are much smaller than printer pixels. To fix this imprecision in line width rendering, PostScript Level 2 added a new feature called *Automatic Stroke Adjustment*. (If you've taken the PostScript Foundations class, we've discussed this feature; check your student notes for more technical details.)

When automatic stroke adjustment is turned on, PostScript minutely adjusts the position of each line against the screen's or printer's pixel grid to give more accurate results in drawing the line. Our array of lines would now paint as in the screen shot at right.



Note that line widths are now correctly rounded to the most accurate number of screen pixels. At a linewidth of 1.5 points, the painted line is rounded up to 2 screen pixels. At 2.5, the line is bumped to 3 pixels, and so forth.

The question is, "How do we activate automatic stroke adjustment in a PDF file?"

I'm glad you asked me that question…

There are two ways: in Distiller using a *prologue.ps* file or in Acrobat using Enfocus' *PitStop* plug-in.

Next Page ->

Distiller and *Prologue.ps*

You can tell Distiller to turn on automatic stroke adjustment in your PDF files by placing a line of PostScript in your *prologue.ps* file. (The following discussion will look familiar if you read the July *Journal;* we used *prologue.ps* in that article, too.)

prologue.ps/epilogue.ps

All versions of Acrobat Distiller will examine the *Data* folder in the Distiller folder for a pair of PostScript files named *prologue.ps* and *epilogue.ps*. If it finds either or both of them, Distiller will execute the contents of *prologue.ps* before each PostScript file it processes and *epilogue.ps* after each file. This allows a PostScript programmer to change the behavior of Distiller.

In our case, we are going to create a *prologue.ps* file that turns on automatic stroke adjustment.

There will be three steps to this process:

1. Create the *prologue.ps* file.
2. Place it in the Data folder for Distiller to find.
3. Tell Distiller to look for *prologue.ps/epilogue.ps* when distilling PostScript files.

Let's do it.

Next Page ->

1. Create prologue.ps

First let's create our *prologue.ps* file.

1. Launch your favorite text editor and create a new file.
2. Type in the following PostScript code:

```
<< /BeginPage { pop true setstrokeadjust } >> setpagedevice
```

   Spacing is not significant here; you can have any number of spaces or tabs between words. Case *is* significant, however; upper and lower case must be exactly as written.

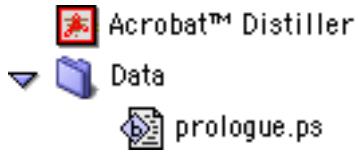
   If you already have a *prologue.ps* file, simply add the above line to those already there.

3. Save the file with the name *prologue.ps.*

   If you used a word processor (Microsoft Word, etc.), rather than a text editor to create this file, make sure you save the file as plain text.

2. Place the File in "Data"

For Distiller to find this file, you must place it in the *Data* folder, located in the same folder as Distiller. Just drag it to the folder.



By the way, the Acrobat 4 documentation says the *prologue.ps* file should be in the same folder as Distiller. This doesn't seem to work; put it in the *Data* folder.

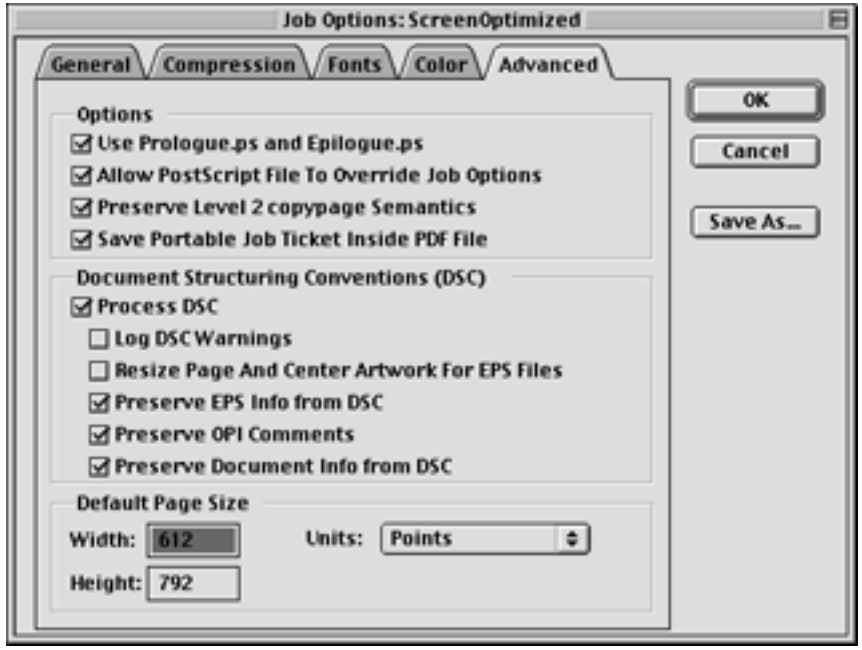
Next Page ->

3. Tell Distiller

Lastly, we need to tell Distiller to look for the *prologue.ps* file.

We do this by going to *Settings>Job Options* and clicking on the *Advanced* tab. The topmost control is a check box labeled *Use Prologue.ps and Epilogue.ps.*

Turn this on.



That's all

From now on, Distiller will apply automatic stroke adjustment to all the PDF files it creates. Your thin lines will all come out appropriately stroked.

Next Page ->

Fixing Line Widths in a PDF File

Up to now, we've talked about creating PDF files in Distiller with Automatic Stroke Adjustment turned on.

But what if it's too late for that? What if you already have a PDF file that displays incorrect line widths?

You can turn on automatic stroke adjustment in a PDF file with my favorite PDF utility: Enfocus' *PitStop.*

Enfocus' *PitStop*

Enfocus Corporation's *PitStop* plug-in for Acrobat is an incredibly handy tool. If you're going to own only one PDF tool, it should probably be PitStop. It's a combination preflighter, editor, and global transmogrifier for PDF files. I have found it endlessly handy over the years.

If you don't have it, you can get a demo from the Enfocus website (www.Enfocus.com) that will let you try out the fix we are discussing here.

PitStop Action Lists

One of PitStop's more impressive features is the ability to create an *Action List,* a set of steps that may be played back as a macro. We are going to create an Action List that turns on automatic stroke adjustment in the current PDF file.

We shall first talk about how to create the Action List and then see how to use it.

Next Page ->

Creating the Action List

Let us first create the Action List. I am *not* going to be explaining the Action List mechanism in detail; it's a long story, the PitStop on-line reference goes into it a bit, and the Acumen Training PitStop class talks about it to completion.

The steps below assume, of course, that you have installed PitStop and have Acrobat running.

1. Open PitStop Action List Control Panel

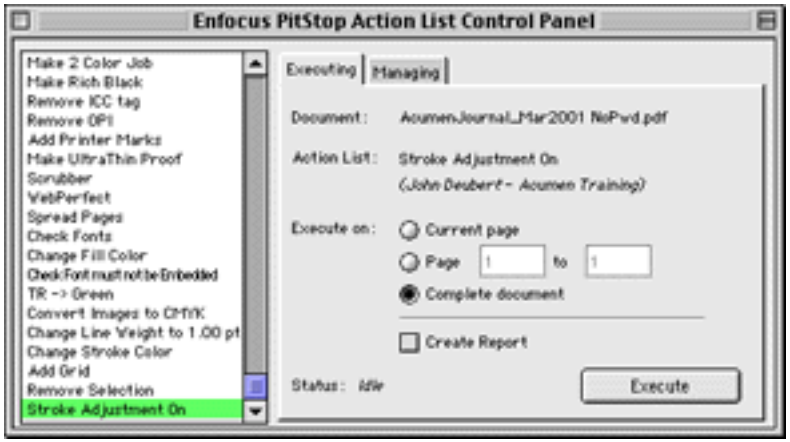
In the Acrobat *Window* menu, select *Show PitStop Action List Panel…*

PitStop will present you with its Action List Control Panel. This dialog box has two tabs:

Executing Here are the controls that let you run an Action List.

Managing Here you create and edit Action Lists.

Down the left side of the control panel is a list of all currently-defined Action Lists.



Next Page ->

2. Go to the Managing Tab

Click on the *Managing* tab in the control panel. You will now be looking at the controls that let you create and edit an Action List.

If we were going to edit an existing Action List, we would select the Action List's name and click the *Edit* menu.

We, of course, are going to create a new Action List.

Next Page ->

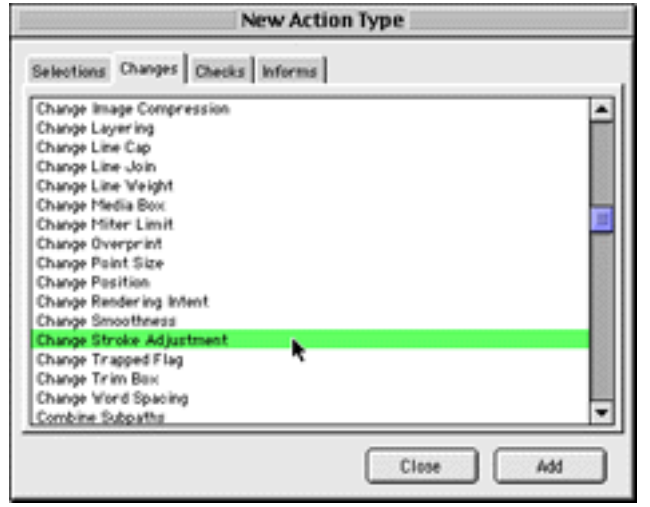
3. Click the "New" Button

PitStop will display the *Action List Editor.*

On the left side of this dialog box is a list, initially empty, of the actions making up this Action List. You add actions to this list by clickin on the "Add" button.

Next Page ->

4. Click the Add button — PitStop will present you with a dialog box displaying all the actions you may add to an Action List.



5. Click the "Changes" tab — Here you get a list of all the changes you may make to a PDF file from within an Action List.

6. Select "Change Stroke Adjustment" — Scroll down to "Change Stroke Adjustment" and click once on the entry.

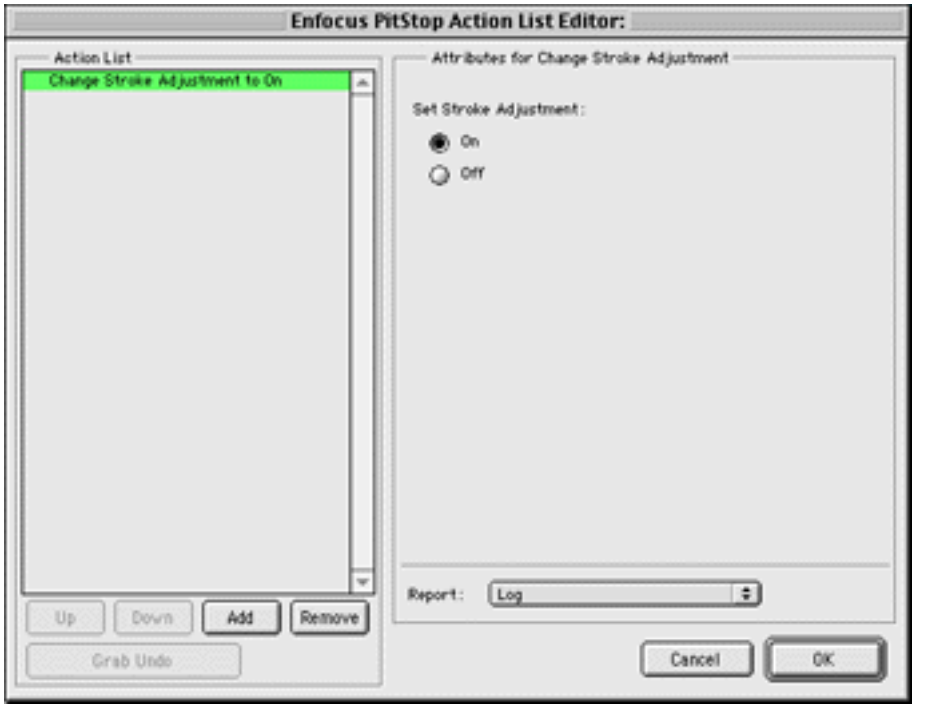
7. Click the "Add" button — This adds the action to the Action List.

8. Click the "Close" button — This closes the dialog box.

Next Page ->

We are looking once again at the Action List Editor dialog box, now showing the single action in our Action List.

Make sure that the *On* radio button to the right is selected.



9. Click the "OK" button

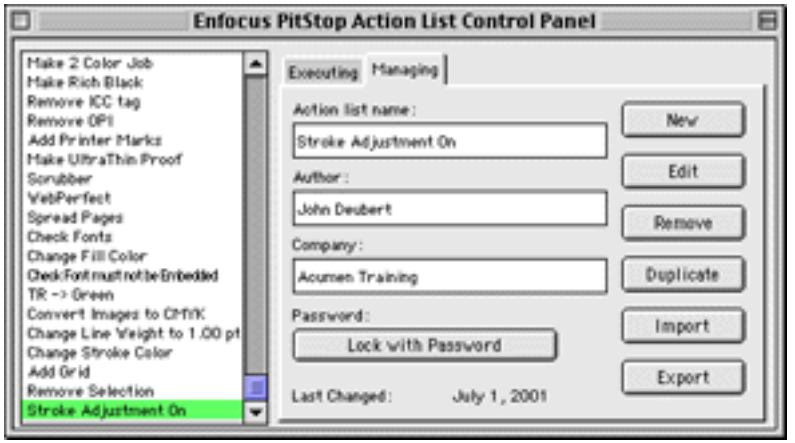
This will return you to the Action List Control Panel.

That's it; we have successfully created an Action List that will fix the too-thick hairlines.

Let's see how we apply this Action List to a PDF file.

Next Page ->

[Next Page ->]



Using the Action List

Using the Action List is pretty easy: Open the PDF file, open the Action List control panel, and click on "Execute." Let's look at this in more detail.

The following steps assume that you've already opened your PDF file in Acrobat.

1. Open Action List Panel

In the Acrobat *Window* menu, select *Show PitStop Action List Panel…*

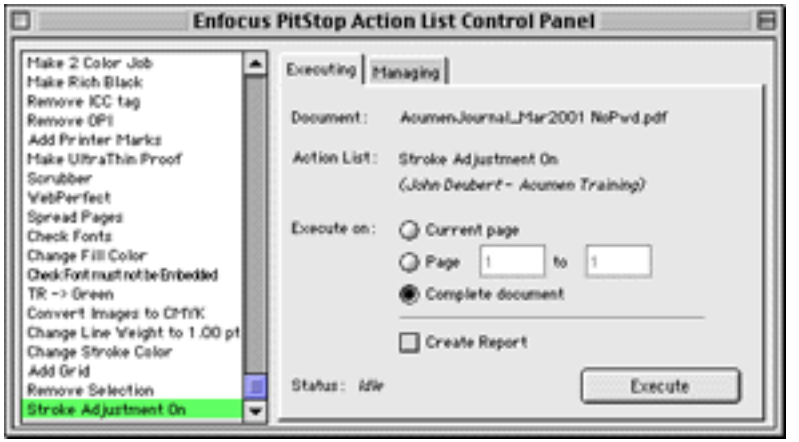
As before, PitStop will present you with the Action List Control Panel.

2. Select the Executing tab

If necessary, select the *Executing* tab. This is the tab from which you run an Action list.

3. Select the Stroke Adjustment Action List

Click once on the Action List we created earlier. (It will probably be at the bottom of the list of Action Lists.)



4. Click the Execute button

PitStop will turn on stroke adjustment in your PDF file. You should see hairlines immediately assume their proper widths.

Next Page ->

Once you have created both the prologue.ps and the PitStop Action List, you should never again be plagued by annoyingly thick hairlines.

Neither of these methods have any particular drawback. Once set up, the prologue.ps runs without any intervention on your part. For those PDF files that already have a hairline problem, I have found that running the Stroke Adjustment Action List is so quick that I hardly notice I'm doing it anymore.

Return to Main Menu

Overriding Character Widths with *Metrics*

One of my first contract programming jobs after I left Adobe Systems (a *long* time ago) was to help a client build a simple variable-data catalog printing system in PostScript. One of the requirements was for a fixed-pitch version of Helvetica. This turned out to be remarkably simple to do, so I thought I'd trot it out for this month's *PostScript Tech.*

Metrics Dictionaries

The PostScript font mechanism provides a means for easily overriding a font's native character widths: a *Metrics* dictionary. Simply place this dictionary in the font dictionary and PostScript will use the character widths it defines.

This month, we'll see how Metrics dictionaries work and how to put one into a font. While we're at it, we'll make a fixed-pitch Helvetica, just for old times' sake.

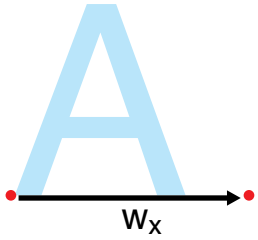
Next Page ->

Metrics Dictionaries

A Metrics dictionary contains key-value pairs that associate character names with metric information for those characters. Specifically, each character name may be associated with one of three things:

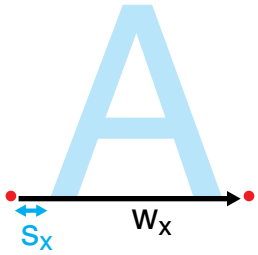
w_x A single number, indicating the horizontal width of the character; that is, the horizontal distance the current point should move after printing the character. The vertical distance and horizontal side-bearing are zero.

This value is expressed in character space units (one-thousandth of an em for Type 1 fonts).



$[\ s_x\ w_x\ ]$ An array of two numbers, indicating the left side bearing (*i.e.,* the white space between the current point and the character) and the character width.

$[\ s_x\ w_x\ s_y\ w_y\ ]$ An array of four numbers, indicating the *x* and *y* side bearings and character widths.



These may be mixed together in a single Metrics dictionary; you need not give the same type of value to all of the character names.

Next Page ->

Making Fixed-Pitch Helvetica

To make a fixed-pitch version of Helvetica, all of whose characters move the current point the same distance, we need to place a Metrics dictionary into the Helvetica font dictionary.

H e l v e t i c a - F i x e d

The keys in this dictionary will be the character names in the Helvetica font; associated with each key we shall have the number 1000, this being the size of an em in a Type 1 font. Thus, each character in our fixed-pitch font will move the current point by an amount equal to the point size.

Changing Fonts

As you remember if you took the PostScript Foundations course, to change a font you must actually create a new font, identical to the original, but containing your changes. There are four steps to this process:

1. Create a dictionary the same size as the original font dictionary.
2. Copy everything from the original font into the new dictionary. (In PostScript Level 1, you must omit copying the */FID* entry; we'll ignore that here and assume we have a PostScript Level 2 or 3 printer.)
3. Make our changes to the new dictionary.
4. Turn the dictionary into a font dictionary with *definefont.*

Let's see how we change our Helvetica.

Next Page ->

Changing Helvetica

Here is the PostScript code for our fixed-pitch Helvetica; the new font is called *Helvetica-Fixed.* (A descriptive name, if not imaginative.)

This PostScript file is on the Acumen Training website's Resources page.

```
/Helvetica findfont           % Get the Helvetica font dictionary
dup length dict begin         % Create a new dict the same length…
currentdict copy              % …and copy Helvetica's k-v pairs into it.

/Metrics                      % Create our Metrics dictionary
     CharStrings dup length dict begin  % Copy all CharString's keys…
     { pop 1000 def } forall  % …into the new dict, with a value of 1000.
     currentdict end
     def

/Helvetica-Fixed currentdict definefont pop  % Turn this dict into a font
end

/Helvetica-Fixed 36 selectfont
0 0 moveto
(Helvetica-Fixed) show

showpage
```

Next Page ->

Step by Step Let's examine this PostScript code in detail.

Create the new dictionary

```
/Helvetica findfont
dup length dict begin
```

We fetch the Helvetica font dictionary, make a copy for later with *dup,* create a dictionary the same length as Helvetica, and move it to the dictionary stack with *begin.* Note that this leaves a copy of the Helvetica font dictionary still on the stack.

Copy Helvetica into the new dictionary

```
currentdict copy
```

We copy the contents of Helvetica (still on the operand stack) into our new dictionary. (Note that *currentdict* returns a pointer to our new dictionary, since that dictionary resides on top of the dictionary stack.)

Create the Metrics dictionary

We want to create a Metrics dictionary with one key-value pair for each character name in the Helvetica font. The best source for all of the character names in Helvetica is the font's *CharStrings* dictionary.

As you recall from your PostScript class (or from reading July's Acumen Journal), the CharStrings dictionary associates character names with the drawing instructions for each character.

Next Page ->

Acumen Journal: PostScript Tech

In making our Metrics dictionary, we shall use *forall* to get each key-value pair out of CharStrings, discard the value (a string full of Type 1 drawing instructions), and replace it with the character width 1000 in our new dictionary

```
/Metrics
```

Put the name */Metrics* on the operand stack.

```
CharStrings dup length dict begin
```

Make a new dictionary the same size as CharStrings and move it to the dictionary stack. Note the *dup;* we will still have a copy of CharStrings on the stack at the end of this line

```
{ pop 1000 def } forall
```

Iterate through CharStrings, discarding the original value associated with each character name and replacing it, in our Metrics dictionary, with the number 1000.

```
currentdict end
def
```

Copy our new dictionary from the dictionary stack to the operand stack, remove it from the dictionary stack, and *def* it as a key-value pair with the name *Metrics.*

Next Page ->

Create the font

```
/Helvetica-Fixed currentdict definefont pop
```

We turn our dictionary into a font dictionary using the *definefont* operator.

```
/FontName <<dict>> definefont  =>  <<fontdict>>
```

Definefont converts a dictionary into a font dictionary with the specified name. It returns a copy of the newly-minted font dictionary. In our case, we discard the return value.

Clean up

```
end
```

Finally, we remove the dictionary (now a font dictionary) from the dictionary stack.

Use the font

```
/Helvetica-Fixed 36 selectfont
0 0 moveto
(Helvetica-Fixed) show
```

Use the font like any other.

Next Page ->

A Couple of Notes

Creating and using a Metrics dictionary is pretty straightforward stuff, once you know it exists. There are just a couple of caveats:

Not with Distill 5

Distiller 5 doesn't honor the character widths in a font's Metrics dictionary. Our sample program will print the text using standard Helvetica character widths.

Distiller 4 and Global Solutions' *PDF Creator* interpret the program correctly.

Appearance Tweaking

Our Helvetica-Fixed left justifies each character within a 1000-unit space. If this looks bad to you, you can specify an appropriate sidebearing for each character. The *forall* loop in our program would become something like:

```
{ pop [ 200 1000 ] } forall
```

I'm not sure that this would look better; Helvetica wasn't designed to be fixed pitch.

Alternatives

Finally, if you need to override character metrics for the printing of only a single string, there are two PostScript operators (added in Level 2) that are more efficient for that purpose. The *xshow* and *xyshow* operators take a string and an array of character widths — *i.e.,* offsets from one character to the next — for each character in the string. The operators print the string, using the character widths provided in the array.

These are well documented in the PostScript Language Reference Manual.

Return to Main Menu

Schedule of Classes, Oct 2001 - Dec 2001

Following are the dates and locations of Acumen Training's PostScript and Acrobat classes. Clicking on a class name below will take you to the description of that class on the Acumen training website. (September has been completely consumed by on-site classes, so there are no Orange County classes this month.)

The PostScript classes are taught in Orange County, California.

PostScript Classes

PostScript Foundations	Orange Co., CA	October 15 - 19	Orange Co., CA	December 10 - 14
Advanced PostScript	Orange Co., CA	November 5 - 9		
PostScript for Support Engineers	Orange Co., CA	October 29 - November 2		
Jaws Development	Orange Co., CA	November 27 - 30		

For more classes, go to www.acumentraining.com/schedule.html

PostScript Course Fees

PostScript classes cost $2,000 per student.
These classes may also be taught on your organization's site.

Registration →
Acrobat Classes →

Acrobat Class Schedule

Acumen training teaches three users' classes in Adobe Acrobat (the links below will take you to the Acumen website's complete description). These are all taught with Acrobat 5, although Acrobat 4 versions may be taught if this is what your site uses.

Acrobat Essentials — This class teaches the student how to make perfect PDF files. It includes complete coverage of the meaning and proper settings of all of the Distiller Job Options.

Interactive Acrobat — Here we show you how to add bookmarks, links, buttons, sounds, movies, form fields, and other interactive features to an Acrobat file.

Troubleshooting with Enfocus' PitStop — This class shows the student how to use all of the capabilities of this popular editing and preflight software.

On-site Only — The Acrobat classes are taught only on corporate sites. If you have an interest in any of these classes for your group, please see the Acumen Training website regarding arranging an on-site class.

Back to PostScript Classes

Return to First Page

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: http://www.acumentraining.com **email:** john@acumentraining.com

telephone: 949-248-1241

mail: 25142 Danalaurel, Dana Point, CA 92629

Registering for Classes

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

Register On-line: http://www.acumentraining.com/registration.html

email: registration@acumentraining.com

telephone: 949-248-1241

mail: 25142 Danalaurel, Dana Point, CA 92629

Back issues

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

Return to First Page

What's New at Acumen Training?

See you at Seybold SF

If you find yourself attending the Seybold conference this month, come by and say "Hi." I'll be teaching three sessions among the "tutorials":

PDF for Prepress

Creating Acrobat Forms

Creating and Fixing PostScript Files.

You can get information about the Seybold seminars and trade show by clicking here.

Return to First Page

Journal Feedback

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

Comments on usefulness. Does the Journal provide you with worthwhile information? Was it well written and understandable? Did you like it, hate it, or did it make you want to eat brussels sprouts? How could we make it better? Do you like the PDF format?

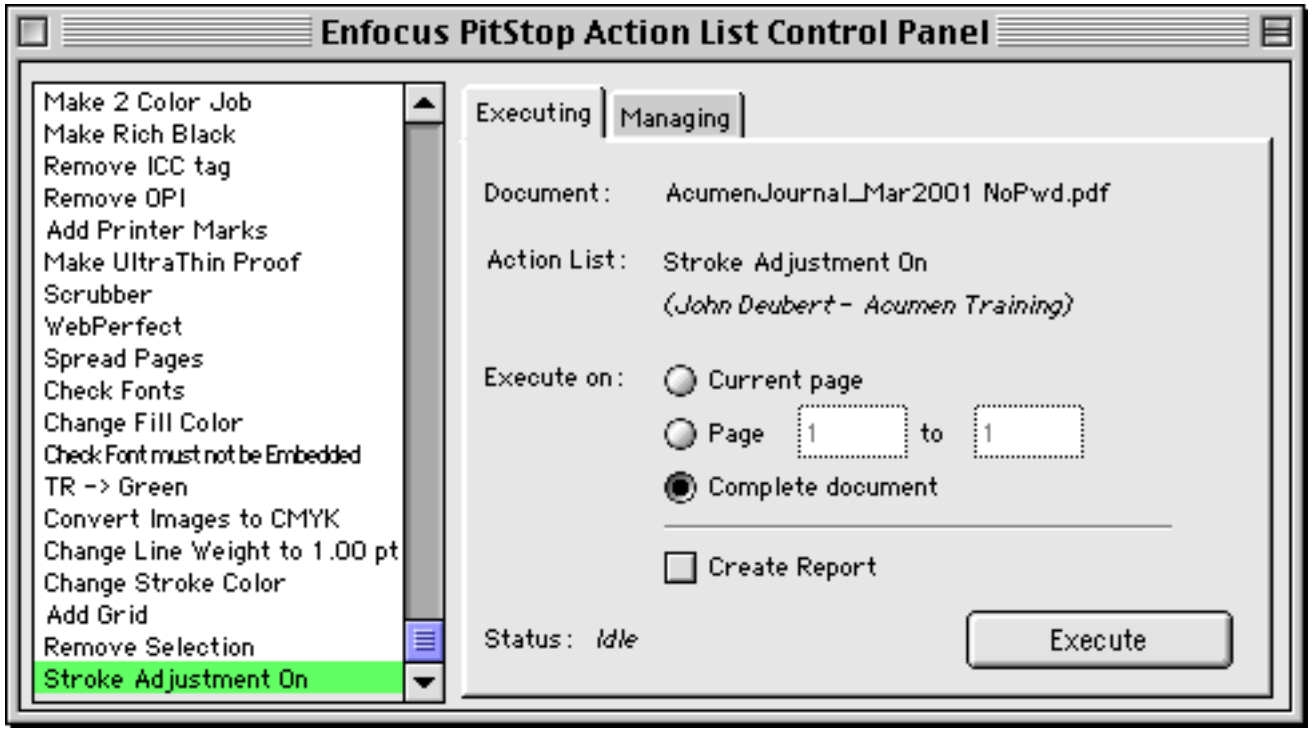
Suggestions for articles. Each Journal issue contains one article each on PostScript and Acrobat. What topics would you like us to address?

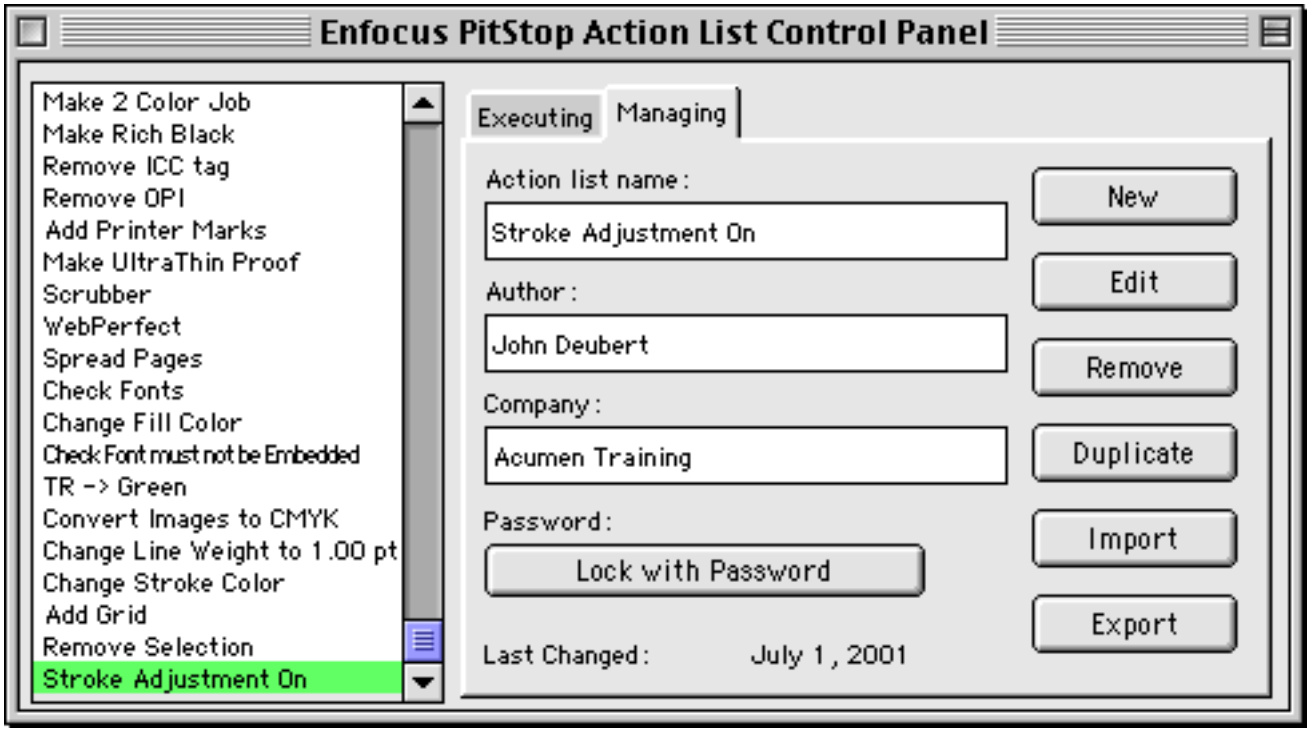
Questions and Answers. We are planning a Q&A section for future issues. Do you have any questions about Acrobat, PDF or PostScript?

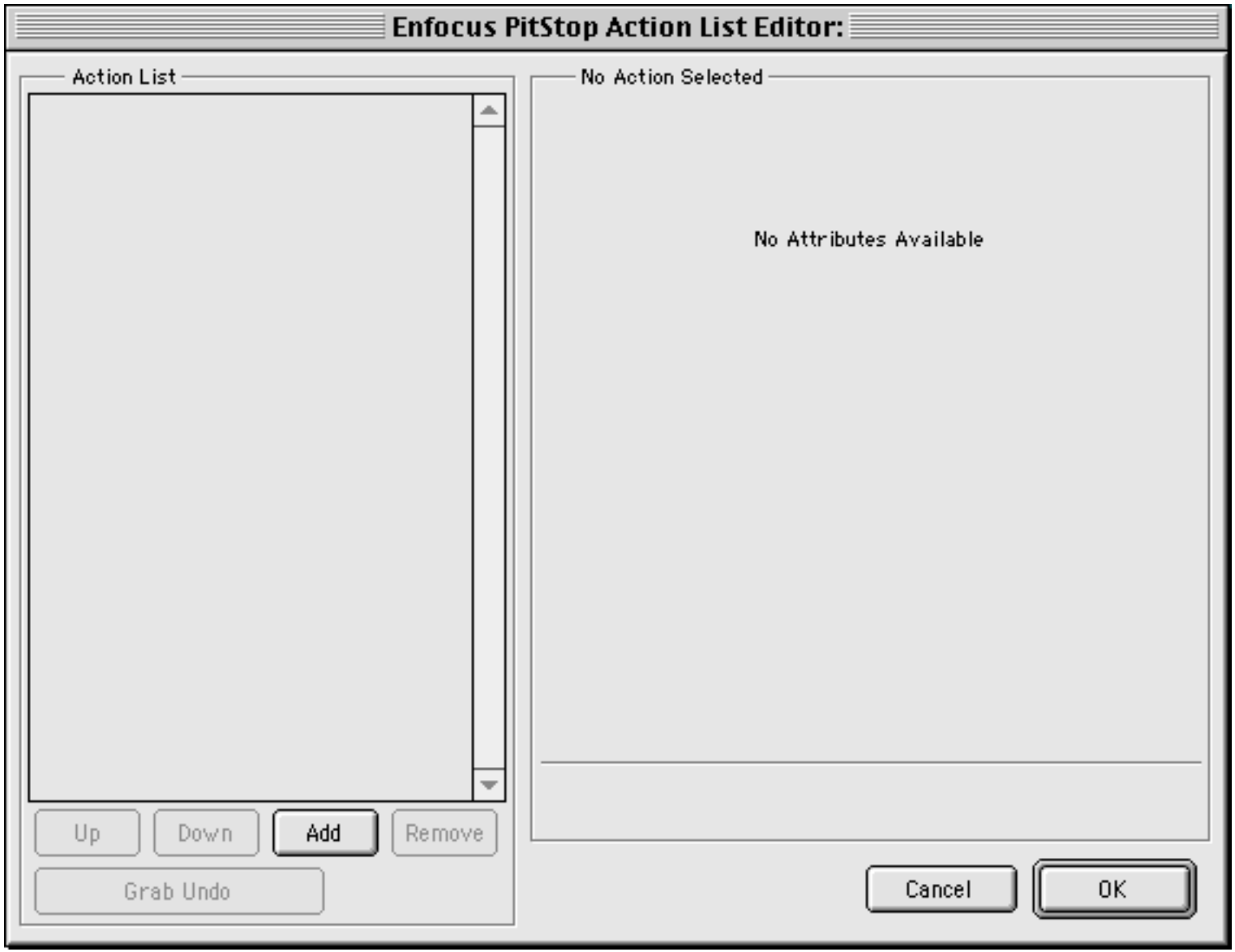
Please send any comments, questions, or problems to:

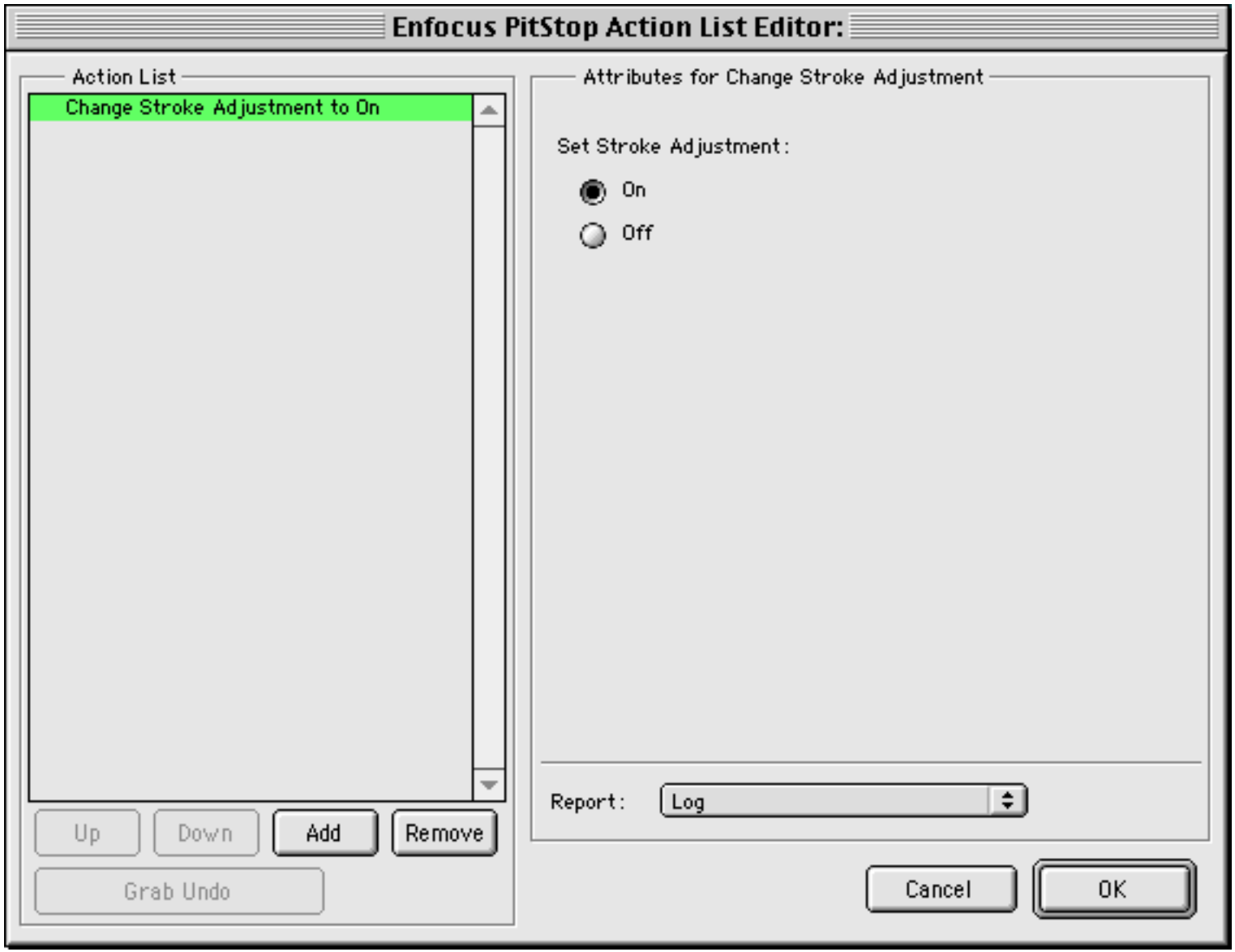
journal@acumentraining.com

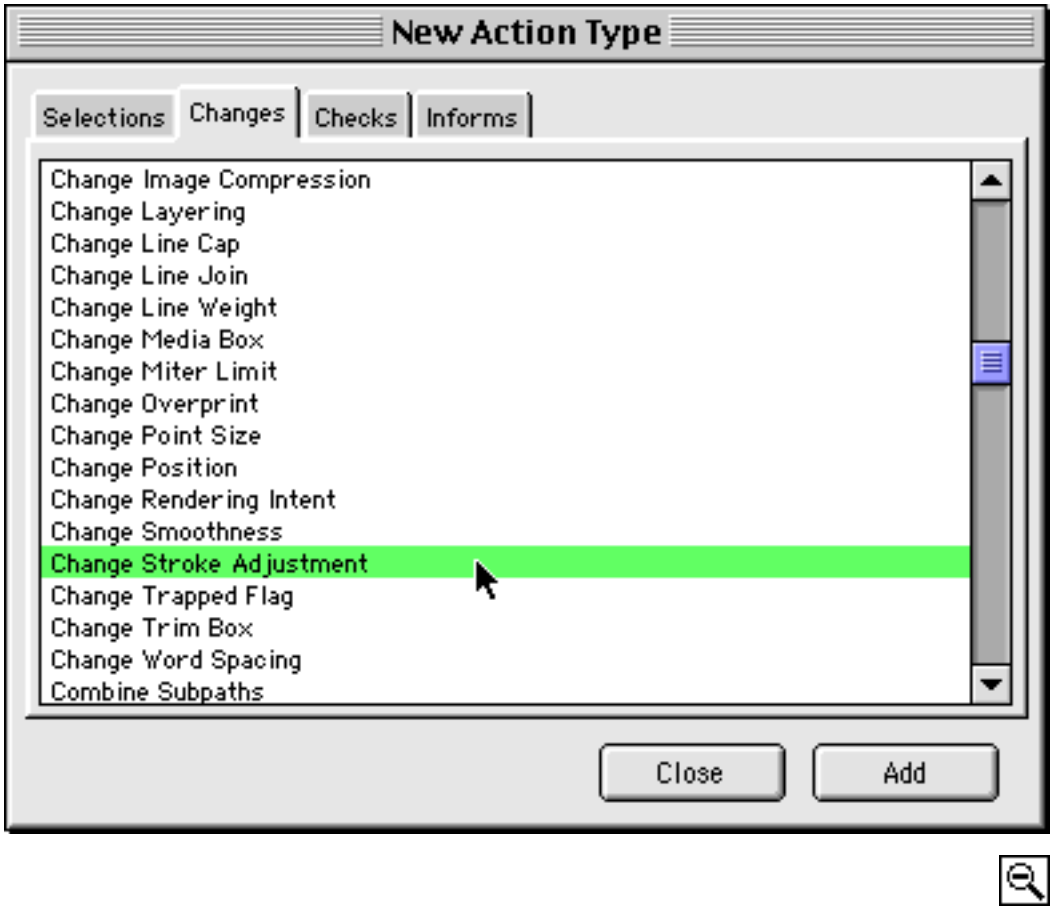
Return to Menu


Enfocus PitStop Action List Control Panel
Make 2 Color Job
Make Rich Black
Remove ICC tag
Remove OPI
Add Printer Marks
Make UltraThin Proof
Scrubber
WebPerfect
Spread Pages
Check Fonts
Change Fill Color
Check Font must not be Embedded
TR -> Green
Convert Images to CMYK
Change Line Weight to 1.00 pt
Change Stroke Color
Add Grid
Remove Selection
Stroke Adjustment On
Executing
Managing
Document: AcumenJournal_Mar2001 NoPwd.pdf
Action List: Stroke Adjustment On
(John Deubert - Acumen Training)
Execute on:
Current page
Page 1 to 1
Complete document
Create Report
Status: Idle
Execute


Enfocus PitStop Action List Control Panel
Make 2 Color Job
Make Rich Black
Remove ICC tag
Remove OPI
Add Printer Marks
Make UltraThin Proof
Scrubber
WebPerfect
Spread Pages
Check Fonts
Change Fill Color
Check Font must not be Embedded
TR -> Green
Convert Images to CMYK
Change Line Weight to 1.00 pt
Change Stroke Color
Add Grid
Remove Selection
Stroke Adjustment On
Executing
Managing
Action list name:
Stroke Adjustment On
Author:
John Deubert
Company:
Acumen Training
Password:
Lock with Password
Last Changed:
July 1, 2001
New
Edit
Remove
Duplicate
Import
Export


Enfocus PitStop Action List Editor:
Action List
No Action Selected
No Attributes Available
Up
Down
Add
Remove
Grab Undo
Cancel
OK


Enfocus PitStop Action List Editor:
Action List
Change Stroke Adjustment to On
Up
Down
Add
Remove
Grab Undo
Attributes for Change Stroke Adjustment
Set Stroke Adjustment:
On
Off
Report:
Log
Cancel
OK


New Action Type
Selections
Changes
Checks
Informs
Change Image Compression
Change Layering
Change Line Cap
Change Line Join
Change Line Weight
Change Media Box
Change Miter Limit
Change Overprint
Change Point Size
Change Position
Change Rendering Intent
Change Smoothness
Change Stroke Adjustment
Change Trapped Flag
Change Trim Box
Change Word Spacing
Combine Subpaths
Close
Add