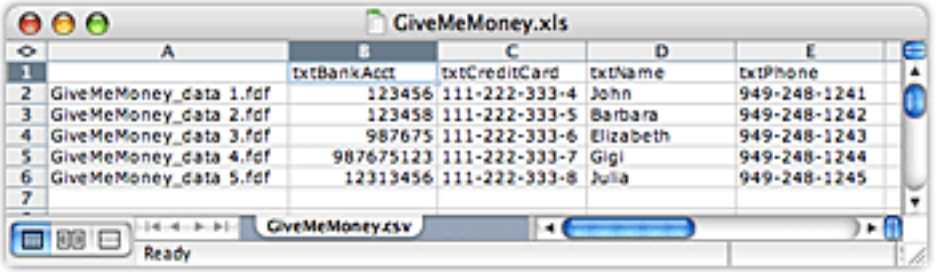
Table of Contents

Exporting Form Data to a Spreadsheet in Acrobat 7

Files on Website

As usual, the files associated with this article are available on the Acumen Training Resources page. Look for the file *ExportToSpreadsheet.zip*.

Welcome to 2005, everyone.



	A	B	C	D	E
1		txtBankAcct	txtCreditCard	txtName	txtPhone
2	GiveMeMoney_data 1.fdf	123456	111-222-333-4	John	949-248-1241
3	GiveMeMoney_data 2.fdf	123458	111-222-333-5	Barbara	949-248-1242
4	GiveMeMoney_data 3.fdf	987675	111-222-333-6	Elizabeth	949-248-1243
5	GiveMeMoney_data 4.fdf	987675123	111-222-333-7	Gigi	949-248-1244
6	GiveMeMoney_data 5.fdf	12313456	111-222-333-8	Julia	949-248-1245
7					

With the new year comes a new version of Acrobat, nicely improved in many ways from Acrobat 6. This month's relatively short article will look at one of Acrobat 7's new features: the ability to convert a series of Acrobat form output files into a single file that can be opened in *Excel* or any other spreadsheet application or database manager.

The feature is pretty easy to use; I thought we'd start the year with something short and simple.

Next Page ->

Background

Submitting Form Data

Consider the simple Acrobat form at right.

The user enters the appropriate information and then clicks the *Pay* button. What happens then?

Somehow, the information the user placed in the form fields must be sent to the organization that distributed the form. This is referred to generically as *submitting* the form data. There are two common ways of doing this:

- The form data can be submitted to the url of a piece of server software that will process the incoming form data and do something useful with it, such as place it in a database.
- The form data can be submitted to an email address, in which case the data will be packaged into an *FDF* file ("form data format") and sent to the specified address as an email attachment.

  FDF is an Acrobat-specific file format intended specifically to contain information from an Acrobat form. It is relatively compact and easily parsed, allowing other software to extract the original form data.

Next Page ->

Acumen Journal: Acrobat User 3

Submitting to a URL is the better way to do things if you are comfortable setting up server-based programs or have an IT department to do such things. For many people, however, the second method allows them to create and distribute an Acrobat form and retrieve responses without having to do additional programming or set up a server-based process.

What you normally do with these FDF files is import them into your copy of the original form. Acrobat populates the form's fields with data from the FDF file. You can then transfer this data—by hand—to your database manager or whatever else you wish.

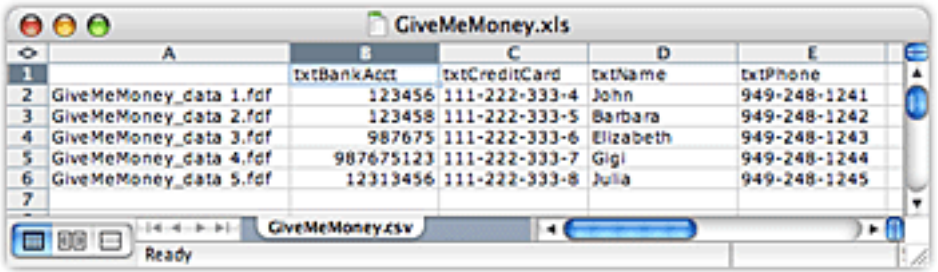
Export to Spreadsheet (or database!)

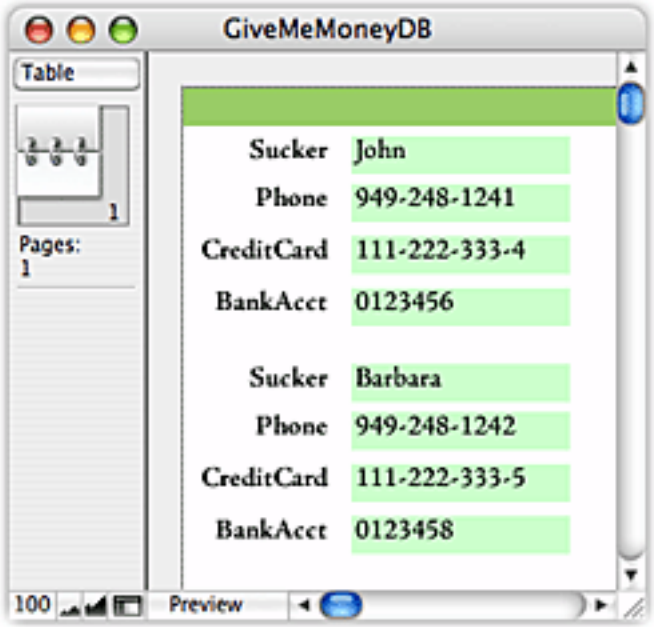
If you retrieve data from a widely-distributed form as FDF files, you will accumulate a possibly large number of these files that must be laboriously imported into Acrobat and the data in them transcribed into your database program. This is tedious at best.

Acrobat 7's *Export to Spreadsheet* feature gives us an alternative. This feature creates a comma-delimited text file out of the form data in a set of FDF files. This comma-delimited file may be opened in a spreadsheet program, such as *Excel.* It may also be imported into most database applications, such as *FileMaker* or *Access.*

This feature makes it easy to transfer Acrobat form data to a database or otherwise use it as you wish.

Next Page ->





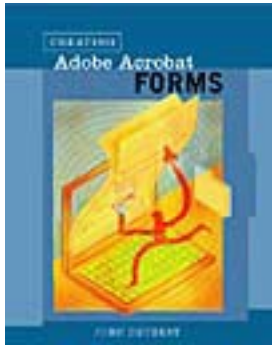
The feature is remarkably easy to use.

Let's look at the steps.

Using *Export to Spreadsheet*

This discussion assumes you have accumulated several FDF files containing information from an Acrobat form you have distributed. Presumably, people have filled out the form and clicked a "Submit" button that you have set up to email the form data, as an FDF file, to you.

(How to create such a form is beyond the scope of this article. Any good book on Acrobat forms—such as my own *Creating Acrobat Forms*—will step you through the process.)

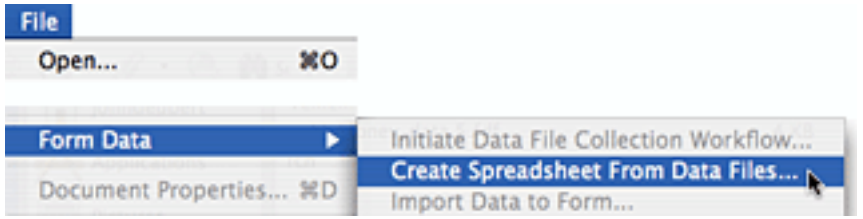


Next Page ->

Creating the Export File

To export these FDF files to a single comma-delimited text file, do the following:

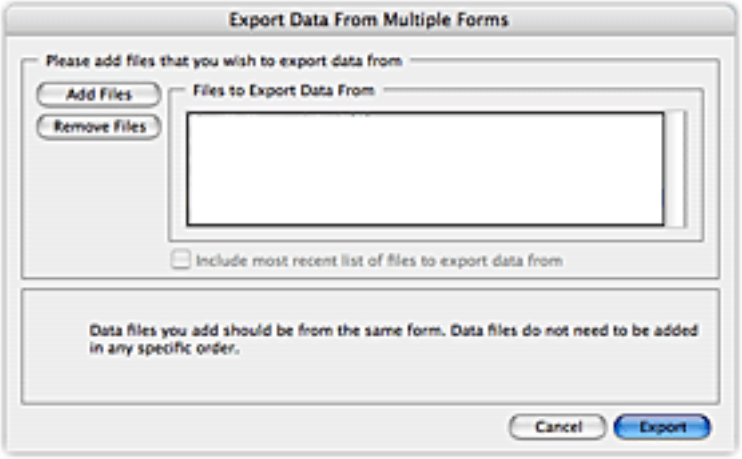
1. Select *File>Form Data>Create Spreadsheet From Data Files...*



   Acrobat will present you with the *Export Data* dialog box, that allows you to create a list of fdf files whose data you want to include in your comma-delimited file.

2. Click on the *Add Files* button.

   Acrobat will present you with a standard Select-a-File dialog box.

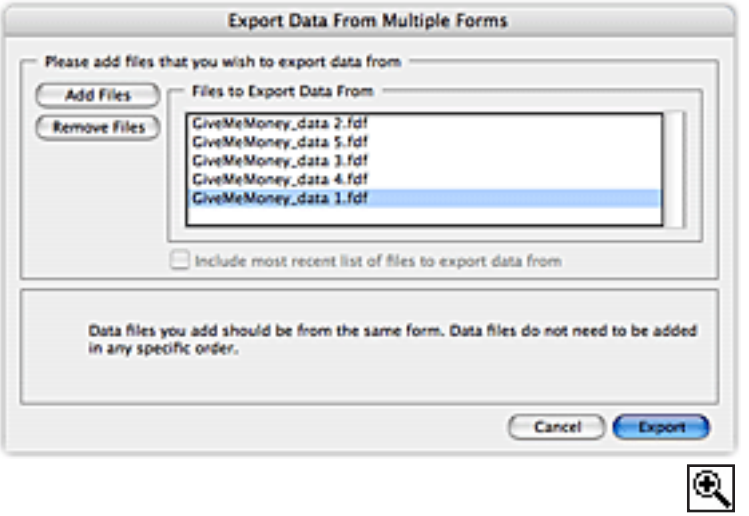


Next Page ->

3. Select the fdf files whose data you want to export to a comma-delimited file.

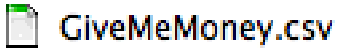
   When you return to the Export Data dialog box, the list will now be populated with the files you selected.

   Note that the order in which the files are entered here has no bearing on the order in which the data is entered in the final export file. As far as I can tell, the data will be added to the comma-delimited file in alphabetical order based on the names of the fdf files.



4. Click on the *Export* button.

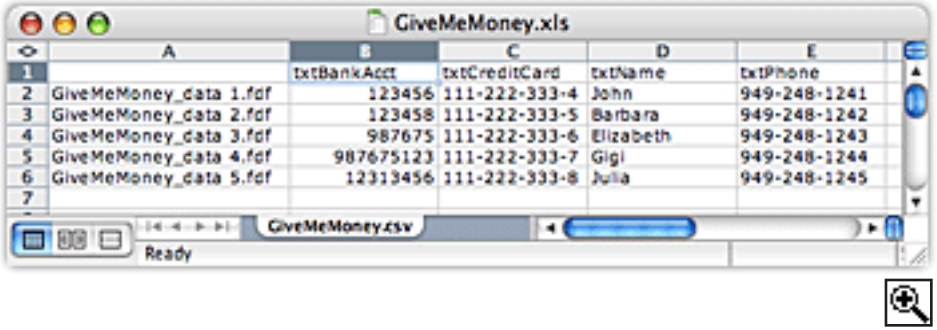
   Acrobat will ask you for the name you want for the output file and then create a file whose suffix is *csv*. This is your comma-delimited spreadsheet file.



Next Page ->

Using the File Spreadsheet Software

The *csv* file may be directly opened in any spreadsheet program. Simply open the file as you would any other spreadsheet document. The result will be as in the illustration at right.



Examining this window, you can see that each row in spreadsheet represents data from one FDF file. The first column in each row contains the name of that row's source FDF file; each of the other columns contains data from one of the form's fields. The first row contains the Acrobat names of the individual form fields.

Database Software

Your database software almost certainly can import comma-delimited text files. The exact procedure will vary from one program to another, but as an example, here's how you would do this in my rather aged FileMaker Pro 5:

You must have already created a database that has fields corresponding to those in your Acrobat form. With that database file open:

1. Select *File>Import Records…*

   Filemaker will present you with a Select-a-File dialog box.

Next Page ->

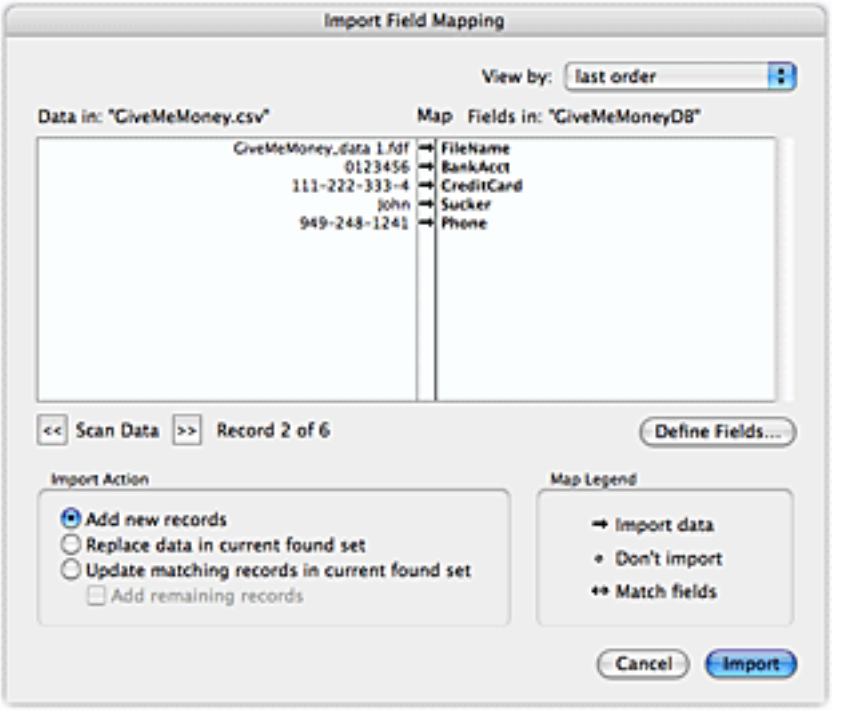
Acumen Journal: Acrobat User 8

2. Select your *cvs* file.

   FileMaker will present you with the dialog box at right, that allows you to indicate the correspondence between the comma-delimited fields in the *cvs* file and fields in the database.

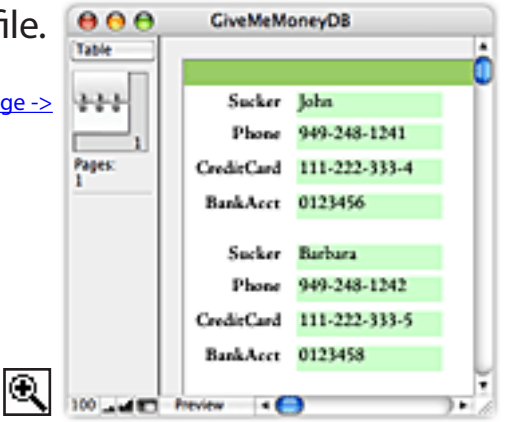
3. Drag the field names up and down the list until the incoming data are correctly aligned with the database's fields.

4. Click the *Import* button.



FileMaker will import the data in the *cvs* file into the database, adding a record to the file for each line of data in the *cvs* file.

Next Page ->



An Important Feature

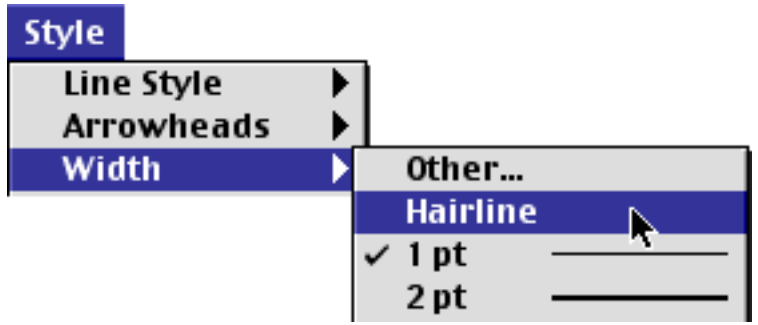
I cannot emphasize how much easier this will make the lives of many—perhaps most—people who use Acrobat forms. Up to now, unless you were comfortable with writing or parameterizing server software, the only way you could use data from Acrobat forms was to tediously import the FDF files into your copy of the form and transcribe the data by hand.

The *Export to Spreadsheet* feature makes Acrobat forms very much more usable for everyone who does not have an IT department on tap.

Return to Main Menu

A Bulletproof Minimum Linewidth

A surprisingly large fraction of the software that produce PostScript files implement their hairlines with a linewidth of *0*. That is, if you select *Hairline* as linewidth, at print time they generate




```
0 setlinewidth
```

As you recall, this causes PostScript to produce a line that is one device pixel thick; this looks perfectly good (very good, in fact) on a low-resolution laser printer, but when sent to an imagesetter or other high-resolution device, the $^{1}/_{3000}$-lines disappear completely.

This can be a problem if you are using this driver-generated code for your own purposes (perhaps concatenating several files together). To prevent hairlines from disappearing, you need to somehow force a minimum linewidth, arranging things so that linewidths less than some threshold value are replaced with that minimum. This is best done with a redefinition of the *setlinewidth* operator.

This is relatively easy, though there is a trick to doing it well. In particular, you need to somehow accommodate the case where the User Space has been severely scaled.

Let's see how to do this.

Next Page ->

Acumen Journal: PostScript Tech 11

Redefining *setlinewidth*

Files on Website

The PostScript code for this article is on the Acumen Training Resources page. Look among the PostScript samples for *setlinewidth.zip*.

This first example is in the zip file as *setlinewidth 1.ps*.

Our first redefinition of *setlinewidth* is relatively simple: simply check whether the argument on the stack is less than some threshold value and, if so, replace the value with that threshold value. Here's the code:

```
/minLineWidth .2 def                % Our threshold value

/setlinewidth                       % Redefine setlinewidth
{   dup minLineWidth lt             % Arg value < minLineWidth?
    { pop minLineWidth } if         % Yes: replace value
    setlinewidth                    % Do a real setlinewidth
} bind def                          % The bind prevents recursion

% Let's try it out:
100 600 translate
5 setlinewidth 0 -125 translate 0 0 200 100 rectstroke
.2 setlinewidth 0 -125 translate 0 0 200 100 rectstroke
.1 setlinewidth 0 -125 translate 0 0 200 100 rectstroke
.05 setlinewidth 0 -125 translate 0 0 200 100 rectstroke
```

If you execute the above code and then examine the output, you will find that the bottom three rectangles actually all have the same linewidth: 0.2 points.

Next Page ->>

So, What's the Problem?

The above redefinition works very well as long as we are working within default User Space. However, we cannot count on this in the general case; if the PostScript code has done a *scale,* then we can get behavior we don't want. For example:

```
.001   .001   scale
1 setlinewidth
stroke
```

In the above code, our redefinition of *setlinewidth* will consider the linewidth, *1,* to be above the minimum. However, since we are scaling down to such a large degree, a linewidth of 1 User Space unit will be extremely thin, far below our minimum linewidth.

A similar problem is possible if the PostScript code scales *up,* as in the code below:

```
100   100   scale
0 setlinewidth
stroke
```

In this case, our *setlinewidth* definition will correctly interpret its argument as being less than the minimum linewidth, but when it eventually sets the linewidth to the substitute value, that new value will be interpreted in the current User Space; in our case a linewidth of, say, *0.2* will yield a line that is 20 points thick.

Next Page ->

Acumen Journal: PostScript Tech 13

How Do We Fix It?

To fix this problem we can convert our linewidths into Device Space and perform all our comparisons there. That is, we shall initially convert our minimum linewidth to Device Space and store the result as our minimum linewidth. Our *setlinewidth* redefinition will convert the requested linewidth to Device Space and compare it to our Device Space minimum linewidth. If it needs to use the substitute, it will convert that minimum linewidth to the *current* User Space and then a call systemdict's *setlinewidth*.

dtransform & idtransform

This solution to our problem is made easy by the use of the *dtransform* and *idtransform* operators.

Δx_{us} Δy_{us} `dtransform` => Δx_{ds} Δy_{ds}

Δx_{ds} Δy_{ds} `idtransform` => Δx_{us} Δy_{us}

These little-noticed operators convert an *x,y* distance between User Space and Device Space. The *dtransform* operator takes a User Space *x* and *y* offset as its arguments and returns the same offset expressed in Device Space. The *idtransform* operator performs the reverse operation, converting a Device Space offset into its User Space equivalent.

We shall use these operators to convert the minimum and requested linewidths between User and Device Spaces.

Next Page ->

Acumen Journal: PostScript Tech 14

Our new *setlinewidth*

Here's the new code:

Files on Website

This second redefinition (followed by some code that tests it, presented later in the article) is in this month's zip file as *setlinewidth 2.ps*.

```
/dsMinLineWidth
     .5 dup dtransform pop    % Cvt min. linewidth to Dvc Space
     def

/setlinewidth    % lw  =>  ---
{    dup                      % Save linewidth (lw) for later
     dup dtransform pop       % Convert lw to Device Space
     dsMinLineWidth lt        % Is it less than our minimum?
     {    pop                 % Yes: discard requested lw…
          dsMinLineWidth      % …& replace with our minimum lw…
          dup idtransform pop % …converted to current User Space
     }if
     setlinewidth             % Now perform an actual setlinewidth
} bind def                    % Do not forget the bind!
```

Step by step

Let's examine this in detail.

`/dsMinLineWidth`

We shall save our minimum linewidth, expressed in Device Space units, in a variable named *dsMinLineWidth*. (The "ds" stands for "Device Space.")

Next Page ->

Acumen Journal: PostScript Tech 15

```
    .5 dup dtransform pop
    def
```

We are going to maintain a minimum linewidth of half a point. Here we convert our minimum value Device Space and use *def* to save the result with the key name *dsMinLineWidth*.

Note that we need to do a *dup* on the linewidth because *dtransform* expects two arguments, an *x* and a *y*. Likewise, the operator returns two values on the stack, of which we need only one; hence, the final *pop*.

```
/setlinewidth          % lw   =>   ---
{    dup                          % => lw lw
```

Our redefinition of *setlinewidth* begins by duplicating the requested linewidth on the stack. If this value is less than our minimum, then we shall discard this number and replace it with our minimum.

```
    dup dtransform pop          % => lw dvc_lw
```

Here we convert our requested linewidth to Device Space. (Note again the odd *dup dtransform pop* idiom made necessary by the way *dtransform* works.)

```
    dsMinLineWidth lt           % => lw bool
```

We compare the requested linewidth, now converted to Device Space, to our minimum linewidth, also expressed in Device Space.

Note that the original linewidth request still resides on the stack, beneath the boolean pushed on the stack by *lt*.

Next Page ->

```
    {    pop                          % => ---
         dsMinLineWidth               % => dvcMin
         dup idtransform pop          % => usrMin
    }if
```

If the requested linewidth is smaller than our minimum, we do three things:

- Discard the original requested linewidth with a *pop.*
- Push the minimum linewidth (in Device Space units) onto the stack.
- Convert this to the current User Space with *idtransform.*

Note that it is important that the minimum linewidth be converted to User Space every time we do a *setlinewidth,* since the actual value we give to *setlinewidth* will be dictated by the state of User Space at execution time.

The *if* clause will leave on the stack the linewidth we want to use. It will either be the original requested value or our minimum value expressed in the current User Space

```
    setlinewidth
} bind def
```

Our redefinition ends by calling *setlinewidth.* Note that the *bind* is absolutely necessary in this definition, since it keeps the final, internal call to *setlinewidth* from recursively calling our redefinition. This final *setlinewidth* will be the one in *systemdict.* (Refer back to your PostScript student notes for a reminder of what *bind* does.)

Next Page ->

Acumen Journal: PostScript Tech 17

But, Does It Work?

Of *course* it does!

Oh, alright; if you want to test it, paste the following code after the redefinition and execute the combined code:

```
% First show that we aren't affecting linewidths over our threshold
100 700 translate
2 setlinewidth 0 0 200 50 rectstroke

% All of the following should print with a linewidth of .5 points
0 -75 translate
.5 setlinewidth 0 0 200 50 rectstroke

0 -75 translate
gsave
.01 .01 scale
1 setlinewidth 0 0 20000 5000 rectstroke
grestore

0 -75 translate
gsave
100 100 scale
.001 setlinewidth 0 0 2 .5 rectstroke
grestore
```

Next Page ->

Acumen Journal: PostScript Tech 18

Final Notes

When Would I Use This?

People who need to force a minimum linewidth are usually doing one of two things:

- Concatenating or otherwise reusing PostScript output from other drivers.
- Importing Encapsulated PostScript code into their own PostScript code.

In both cases, they are using PostScript code taken from more-or-less arbitrary sources. Such PostScript code can do all sorts of strange and wondrous things, including specifying excessively small linewidths.

Redefinition Comes *first!*

It is important that this redefinition of *setlinewidth* come before pretty much anything else in your PostScript stream. In particular, it is important that the currently-available *setlinewidth* at the time you do your redefinition is the systemdict-resident operator definition. Otherwise, you risk having the internal call to *setlinewidth* be recursive.

Just make sure the redefinition precedes the EPS code or driver output with which you are working and all will be well.

Thanks, Charly!

Charly Tischler of Océ Printing Systems GmbH pointed out the embarrassingly long-standing weakness in my usual *setlinewidth* redefinition that led to this article.

Ah, well. Better smart late than never. Thanks, Charly.

Return to Main Menu

Schedule of Classes, Jan - Apr 2005

Following are the dates of Acumen Training's upcoming PostScript and PDF Technical classes. Clicking on a class name below will take you to the description of that class on the Acumen training website.

These classes are taught in Orange County, California and on corporate sites world-wide. See the Acumen Training web site for more information.

Technical Classes

PDF File Content and Structure			Mar 21–24
PostScript Foundations	Jan 31–Feb 4		
Variable Data PostScript			
Advanced PostScript			Mar 7–11
PostScript for Support Engineers			Apr 4–8
Jaws Development		*On-site only*	

Course Fee The PostScript and PDF classes cost $2,000 per student.

Registration Info

Acrobat Class Schedule

These classes are taught occasionally in Costa Mesa, California, and on corporate sites. Clicking on a course name below will take you to the class description on the Acumen Training web site.

Acrobat Essentials — *No Acrobat classes scheduled for this quarter.* See the Acumen Training website regarding setting up an on-site class.

Interactive Acrobat

Creating Acrobat Forms

Acrobat Class Fees — *Acrobat Essentials* and *Creating Acrobat Forms* (½-day each) cost $180.00 or $340.00 for both classes. There is a 10% discount if three or more people from the same organization sign up for the same class.

Registration ->

Return to Main Menu

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: 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: http://www.acumentraining.com/registration.html

email: registration@acumentraining.com

telephone: 949-248-1241

mail: 24996 Danamaple, Dana Point, CA 92629

Back issues

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

Return to First Page

What's New at Acumen Training?

New PDF Class

Nothing too new this month. I am still in the early stages of laying out the second PDF File Content and Structure class. The topic list is below; as before, if you think something should be added to or dropped from this list, send an email to john@acumentraining.com.

Preliminary Topic List

- Overprinting
- CID Fonts
- Halftones
- Marked Content
- Rendering Intents
- Smooth shading
- Reference XObjects
- Cross reference streams
- BX & EX
- File Spec
- Masked Images
- Digital Signatures
- AcroForm
- Transfer Functions
- Shape dictionaries
- Layers
- Name Dictionaries
- Patterns
- Composite Fonts
- Linearized PDF
- Stroke Adjustment
- Halftones
- Text Knockout
- Object streams
- More on data structures

Return to First Page

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 remind you of your old Uncle Jasper, who would always drone on and on about some incident in his youth involving a goat and a ball of twine and yet never quite finished the story?

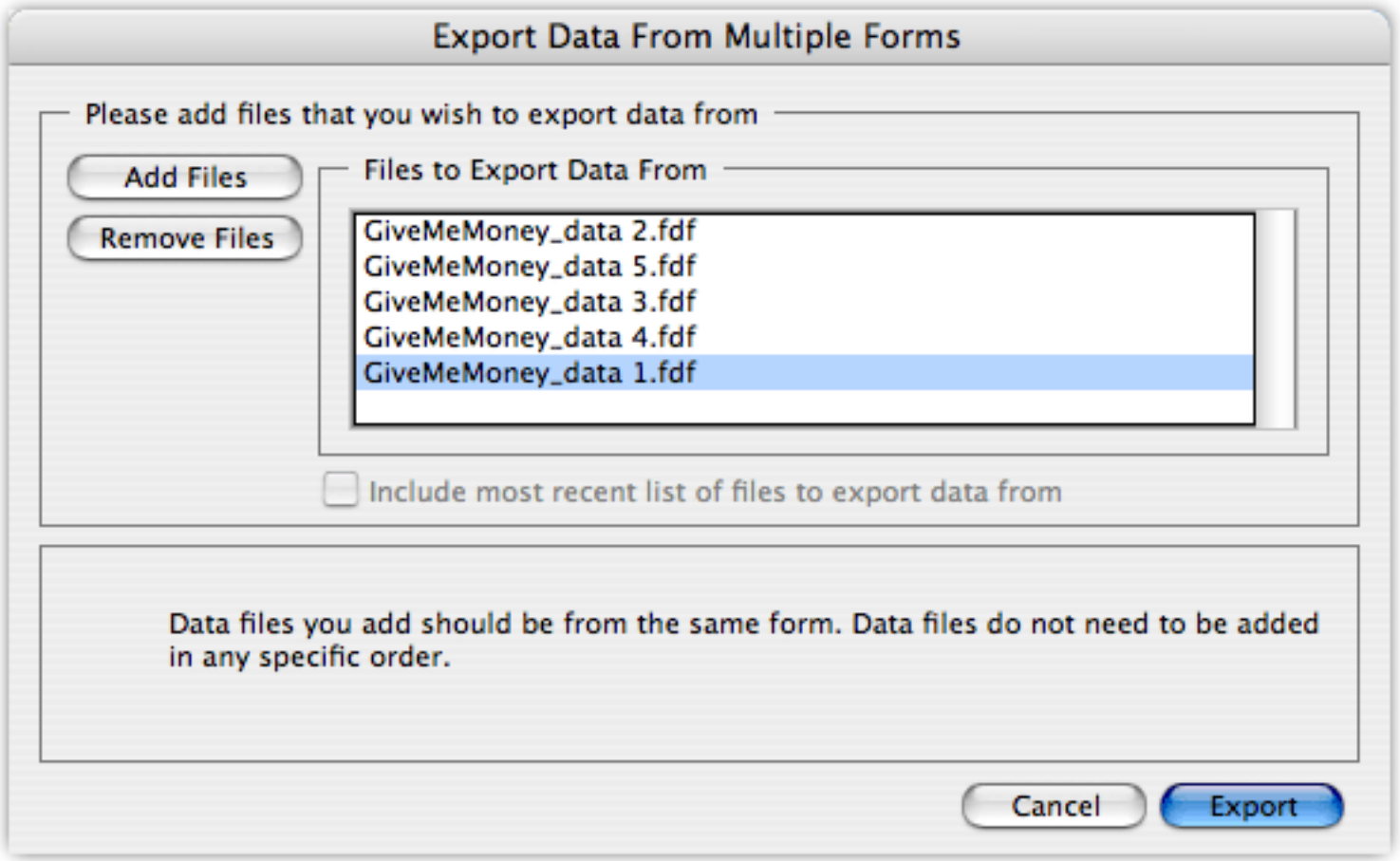
Suggestions for articles. Each Journal issue contains one article each on PostScript and Acrobat. What topics would you like me to write about?

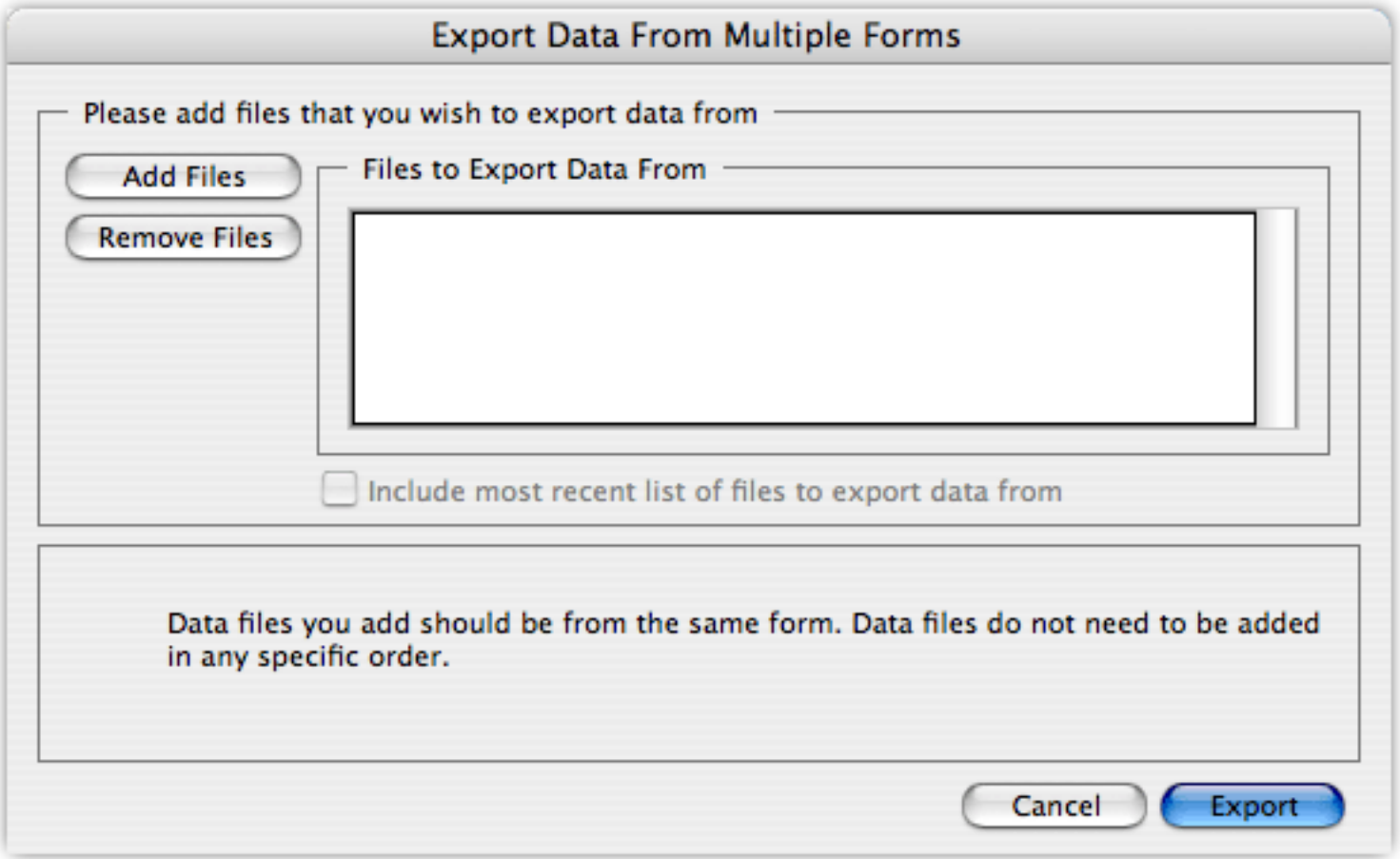
Questions and Answers. Do you have any questions about Acrobat, PDF, or PostScript? Feel free to email me about. I'll answer your question if I can. (If enough people ask the same question, I can turn it into a Journal article.)

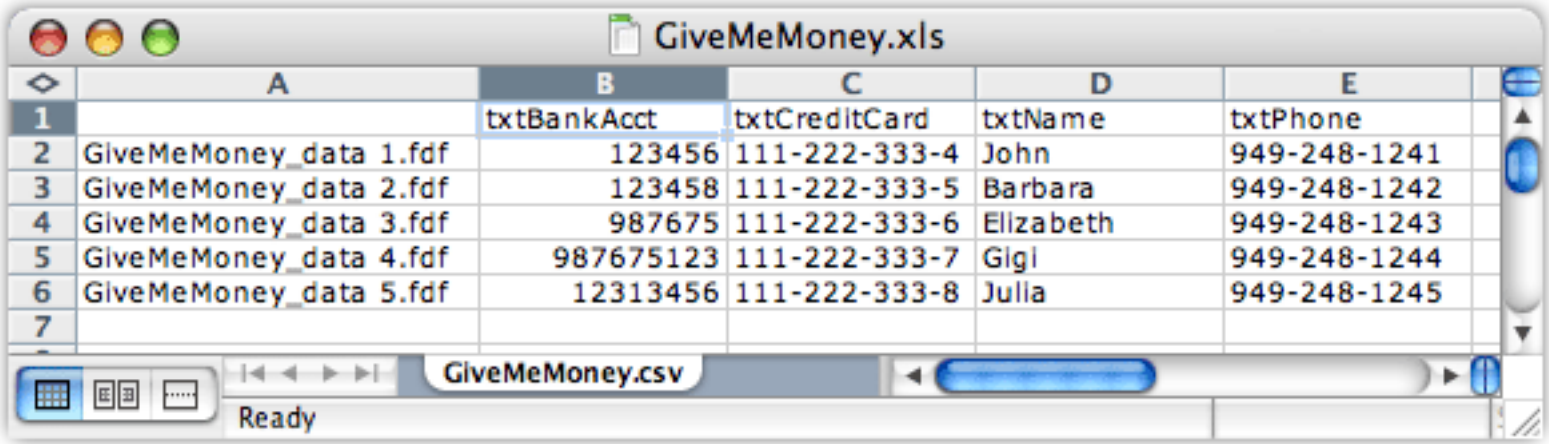
Please send any comments, questions, or problems to:

journal@acumentraining.com

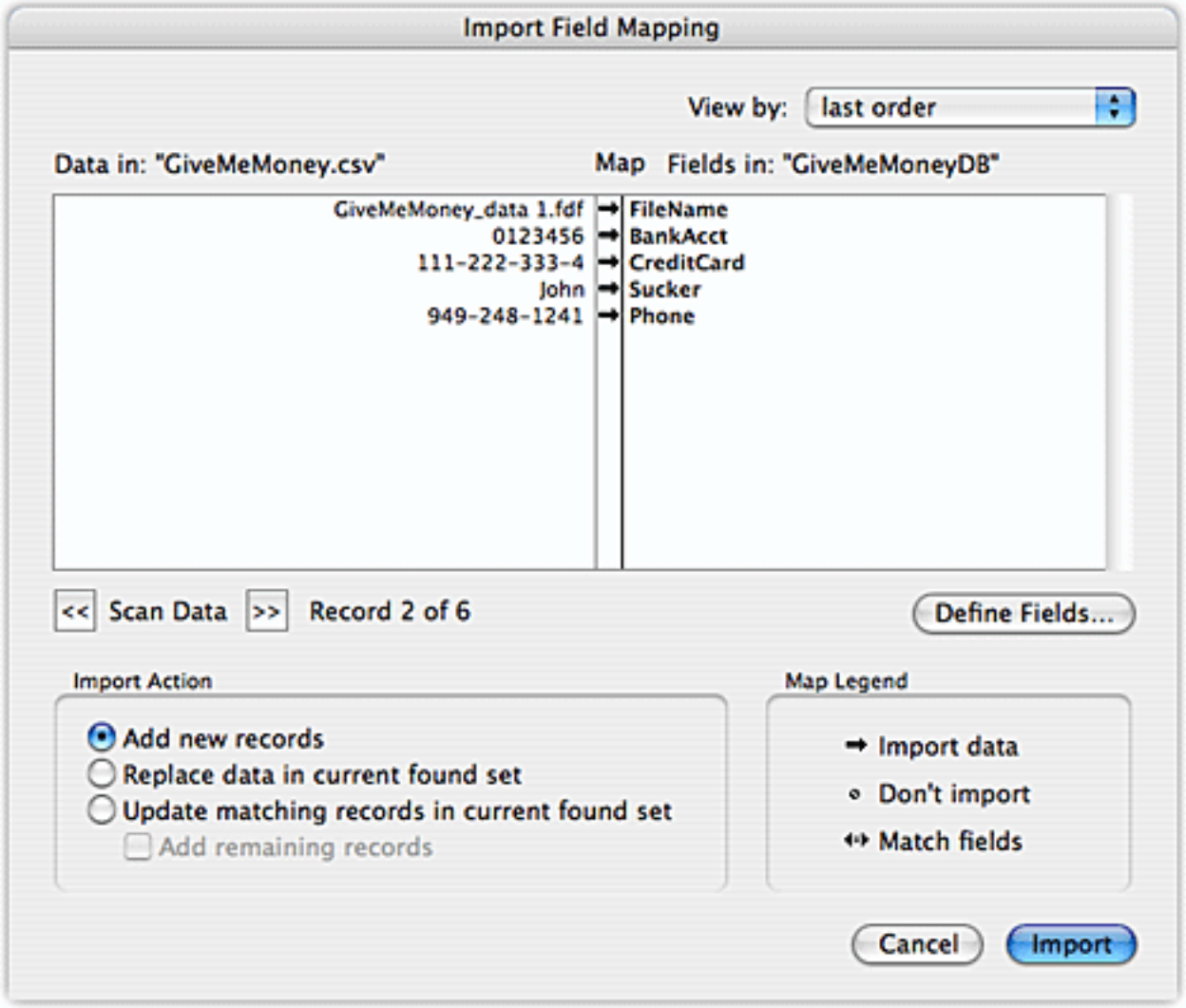
Return to Menu

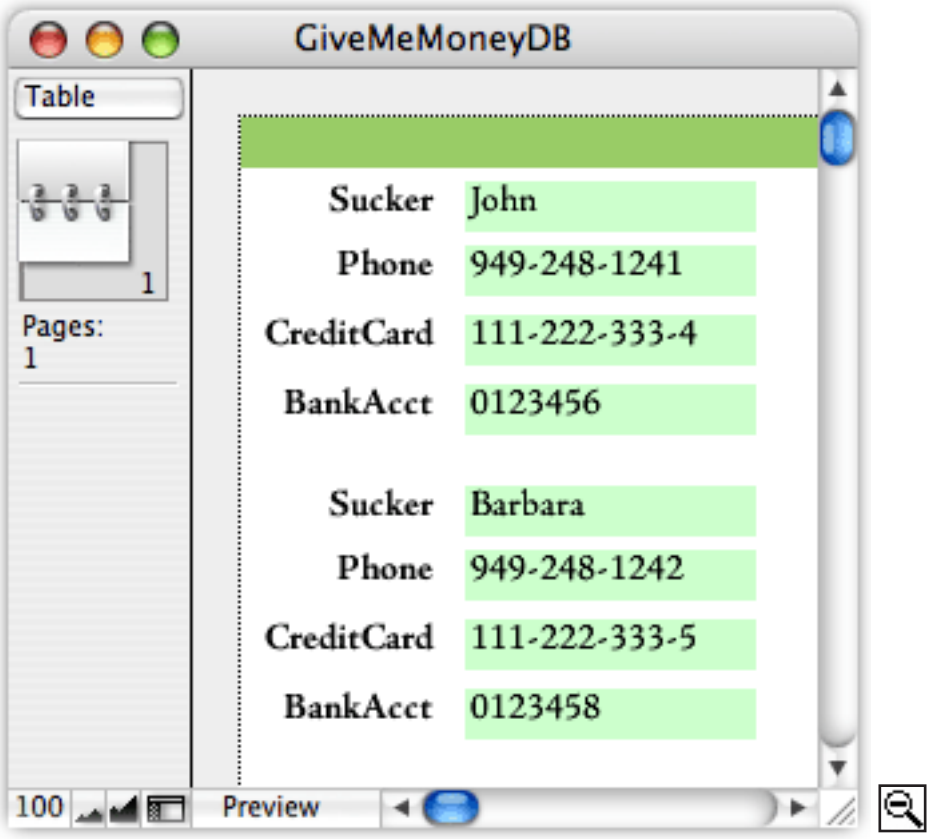

Export Data From Multiple Forms
Please add files that you wish to export data from
Add Files
Remove Files
Files to Export Data From
GiveMeMoney_data 2.fdf
GiveMeMoney_data 5.fdf
GiveMeMoney_data 3.fdf
GiveMeMoney_data 4.fdf
GiveMeMoney_data 1.fdf
Include most recent list of files to export data from
Data files you add should be from the same form. Data files do not need to be added in any specific order.
Cancel
Export


Export Data From Multiple Forms
Please add files that you wish to export data from
Add Files
Remove Files
Files to Export Data From
Include most recent list of files to export data from
Data files you add should be from the same form. Data files do not need to be added in any specific order.
Cancel
Export


GiveMeMoney.xls
A
B
C
D
E
1
txtBankAcct
txtCreditCard
txtName
txtPhone
2
GiveMeMoney_data 1.fdf
123456
111-222-333-4
John
949-248-1241
3
GiveMeMoney_data 2.fdf
123458
111-222-333-5
Barbara
949-248-1242
4
GiveMeMoney_data 3.fdf
987675
111-222-333-6
Elizabeth
949-248-1243
5
GiveMeMoney_data 4.fdf
987675123
111-222-333-7
Gigi
949-248-1244
6
GiveMeMoney_data 5.fdf
12313456
111-222-333-8
Julia
949-248-1245
7
GiveMeMoney.csv
Ready

FileMaker Pro 5's *Field Mapping* Dialog Box




GiveMeMoneyDB
Table
1
Pages:
1
Sucker John
Phone 949-248-1241
CreditCard 111-222-333-4
BankAcct 0123456
Sucker Barbara
Phone 949-248-1242
CreditCard 111-222-333-5
BankAcct 0123458
100
Preview