Table of Contents

Journal feedback: suggestions for articles, questions, etc.

Idiom Recognition

Background Reading

This article uses the PostScript resource mechanism. If you don't remember how resources work, you might want to reread the January and April 2007 issues of the *Acumen Journal*.

Consider Adobe's justifiably proud release of PostScript Level 2, back in 1991. This amazingly rich extension to the PostScript language promised to print up to 5 times faster than Level 1. Unfortunately, this promise remained unfulfilled for two to three years after Level 2's release, because all the applications and drivers were still producing Level 1 PostScript code. It doesn't matter how much better PostScript Level 2 is if no one is using its features. Many end users at the time didn't see any benefit to PostScript Level 2 and considered Adobe's talk of improvements to be marketing exaggeration.

Adobe was determined that this wouldn't happen when they released PostScript Level 3. They wanted to ensure that users would see an immediate benefit to buying a Level 3 printer even if they were still printing with Level 2 PostScript code.

The mechanism by which this would happen is *Idiom Recognition*. This Level 3 feature allows a printer to recognize known pieces of PostScript Level 2 code (perhaps the procedure that QuarkXpress 5 used to print gradients, for example) and replace them with equivalent—but much faster and better—Level 3 code.

Although this feature was intended to be used by printer engineers, letting their printers update old-fashioned PostScript code to use newer language features, it is occasionally useful to a PostScript programmer as a way to "patch" legacy code, usually generated by in-house applications.

Let's see how it all works.

IdiomSet Resources

Level 3's *bind* Operator

The part of PostScript Level 3 that does the actual code substitution is the *bind* operator. As you know, this operator takes a procedure body as its argument:

```
/inch { 72 mul } bind def
```

The *bind* operator replaces all operator names within the procedure body with the corresponding operator definitions. In PostScript Level 3, *bind* does something in addition: it compares the procedure against a set of known "problem procedures"; if it finds a match, it pops the original procedure off the stack and replaces it with a pre-defined (presumably Level 3) alternative. By design, the alternative procedure takes the same arguments and carries out the same graphic task as the original procedure, but does so using faster, better-looking Level 3 code.

The replacement procedure becomes the value of *def's* key-value pair.

In PostScript parlance, the original, problematic procedure is called the *template* and its replacement is called the *substitute*.

IdiomSet Resources

Template-Substitute pairs are defined by an *IdiomSet* resource. An IdiomSet consists of a dictionary containing an arbitrary number of named arrays, each containing two bound procedures: the template and its substitute; generically, its definition looks like this:

```
/IdiomSetName     <<
     /ArbitraryName1   [
          { template } bind
          { substitute } bind
     ]

     /ArbitraryName2   [
          { template } bind
          { substitute } bind
     ]
>> /IdiomSet defineresource
```

The names of the arrays within the IdiomSet dictionary are never used and so may be anything you like; something descriptive would be nice.

Default IdiomSets

Every PostScript RIP ships with IdiomSet resources appropriate to the market for which the printer is intended. A printer aimed at the pre-press market would ship with IdiomSets that convert selected PostScript procedures used by QuarkXpress, Illustrator, PageMaker (remember PageMaker?), etc. A printer intended for use in a business office would supply IdiomSets that patch Word and Excel output.

Distiller 9, for example, ships with IdiomSets that have these self-explanatory resource names:

QuarkBoxes	TexDvipsFixes	AGMPrintFixes
WinDrvFixes	PageMaker5044	SAPPrintFixes
WinDrvBoldText		

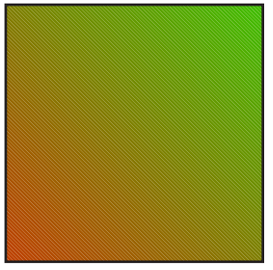
Defining an IdiomSet

As an example, consider the following PostScript code that defines a *GradientFill* procedure that, when executed, fills the current path with a gradient:

```
/dvcRes 72 def                          % The device resolution
/PointsPerDot 72 dvcRes div def

/GradientFill     % x0 y0  x1 y1  r0 g0 b0  r1 g1 b1 => ---
{    gsave
     clip

     3 index sub /dBlue exch def     % Record the amount the rgb will change
     3 index sub /dGreen exch def    % across the gradient
     3 index sub /dRed exch def

     /b exch def                     % r, b, & b are the current values of those
     /g exch def                     % color components
     /r exch def
```

```
3 index 3 index translate       % Stack: x0 y0  x1 y1

% Calculate delta-x & -y
3 -1 roll sub           % x0 x1 dy
3 1 roll sub neg        % dy dx
exch                    % dx dy

% From this, get the angle of the gradient's axis & rotate by that angle
2 copy exch atan        % atan wants:  num  denom
rotate                  % Stack: dx dy

dup mul exch            % Calculate the length of the gradient's axis
dup mul add             % Just using good ol' Pythagoras here
sqrt
/t1 exch def

% Calculate the various iteration amounts
/nSteps t1 PointsPerDot div def
/dRed dRed nSteps div def
/dGreen dGreen nSteps div def
/dBlue dBlue nSteps div def

0 PointsPerDot t1       % Draw a bunch of 1-pixel-thick rectangles
{
    r g b setrgbcolor
    -500 1 1000 rectfill
    /r r dRed add def
    /g g dGreen add def
    /b b dBlue add def
} for
```

```
    grestore      % Put the coordinate system (& everything else) back
    newpath       % Erase the current path
} bind def

% Now, we construct a path and fill it with out new procedure
250 350 moveto
100 0 rlineto 0 100 rlineto
-100 0 rlineto closepath
200 300  400 500  1 0 0  0 1 0 GradientFill

showpage
```

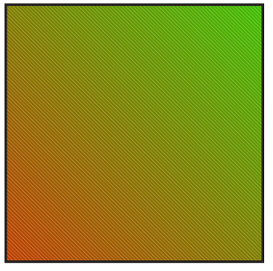


This is a typical Level 2 way of doing gradient fills: clip to the current path and then paint a series of rectangles, each a slightly different color than the next. This works well enough, but it is either slow (we are doing a very large number of 1-pixel-wide rectangles), chunky (if we are doing a smaller number of *n*-pixel-wide rectangles), or both, depending on circumstance.

The *GradientFill* procedure above takes ten numbers as its arguments: the *x* and *y* coordinates of the endpoints of the gradient's axis and the *r,g,b* of the endpoint colors. It would not be particularly difficult to write a procedure that takes these same arguments and paints a gradient using the Level 3 *shfill* operator. In fact, here is the alternative version:

```
/GradientFill2              % x0 y0  x1 y1  r0 g0 b0  r1 g1 b1 => ---
{   gsave
    clip

    8 dict begin            % Assemble a Type 2 shading dictionary
    /ShadingType 2 def
    /ColorSpace [ /DeviceRGB ] def
    5 dict begin            % Create the function dictionary
        /FunctionType 2 def
        /Domain [ 0 1 ] def
```

```
            /N 1 def
            3 array astore          % r1 g1 b1 become the C1 array
            /C1 exch def
            3 array astore          % r0 g0 b0 become the C0 array
            /C0 exch def
        /Function currentdict end def
        4 array astore              % x0 y0 x1 y1 go into the Coords array
        /Coords exch def
        currentdict shfill
        end
        grestore
        newpath
} bind def
```



Not only is this version of the procedure shorter than the original, we don't need to build into the code any resolution dependencies. The *shfill* operator will render the gradient at the device's resolution, whatever that might be.

Note that this procedure is a pin-for-pin replacement of the original; it takes the same arguments and produces the same gradient on the page—only faster and better.

So, how do we tell Level 3 to replace the original *GradientFill* with the improved one whenever it encounters it?

By creating an IdiomSet resource.

Creating an IdiomSet

Here's the code, somewhat bowdlerized:

```
<< /IdiomRecognition false >> setuserparams      % Turn off idiom recognition
/RealBlends
<<   /Blends
     [    {    gsave                  % Template procedure
               clip
               3 index sub /dBlue exch def
               ...
               0 PointsPerDot t1
               {
                    r g b setrgbcolor
                    ...
               } for
               grestore
               newpath
          } bind

          {    gsave                  % Substitute procedure
               clip
               8 dict begin
               /ShadingType 2 def
               ...
               currentdict shfill
               (Replacement) =        ...
          } bind
     ]
>> /IdiomSet defineresource
<< /IdiomRecognition true >> setuserparams       % Turn idiom recognition on, again
```

why setuserparams?

This is reasonably straightforward code, except for the mysterious calls to *setuserparams* before and after the call to *defineresource*. These turn idiom recognition off (before defining the IdiomSet resource) and back on again (afterward).

We need to do this because the template and substitute procedures in the IdiomSet resource must be bound; however, it is the *bind* operator itself that does the procedure substitution. To prevent *bind* from attempting to carry out a substitution while we are trying to create the IdiomSet resource, we turn idiom recognition off for the duration.

So now what?

Having defined the RealBlends IdiomSet resource, anytime the *bind* operator sees the original, Level-2-style blend procedure,

```
/GradientFill { old procedure } bind
```

it will discard it and replace it with the new, improved Smooth Shading procedure.

```
/GradientFill { new procedure } bind
```

The *def* operator creates a named procedure using our Level 3 code and whenever the driver's PostScript output executes *GradientFill,* it will execute that Level 3 code.

Importantly, what the user will see is that blends look nicer and print more quickly on their nice, new Level 3 printer.

Sample on Web Site

A sample file that shows our RealBlends IdiomSet in action is on the Acumen Training Resources page. Look for *IdiomSet.ps*.

End Notes

What's it good for?

Not too long ago, I had a question from a group that had just been given the job of improving the printing speed of their in-house variable-data application. Said app had been written years before, was using a mix of Level 2 and Level 1 techniques (and hadn't been written very well in the first place). The original programmers, of course, were long gone from the company.

The application's original C code was, as is traditional, poorly documented, so the new caretakers really didn't want to rewrite the PostScript-generating part of their application. Since they printed to a known stable of in-house printers, they were able to create a series of IdiomSet resources behind the RIPs' Server Loops that fixed up the old PostScript, replacing several bottleneck procedures with new, snazzy code.

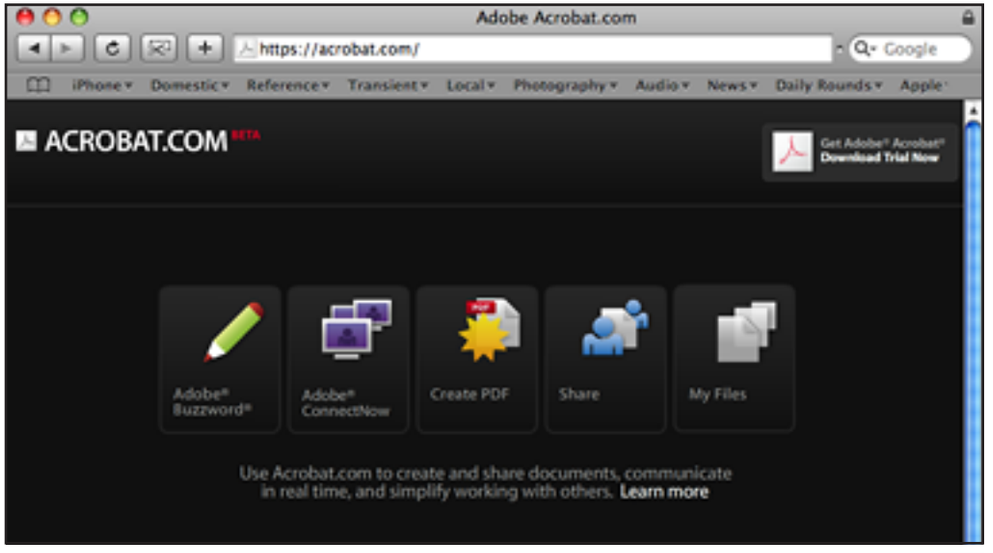
In the end, they greatly improved the speed of their output without having to rewrite any of the application's source code.

Any Problems With This?

None significant. Certainly *bind* slows a bit, since it must compare its procedure argument to all existing templates. However, this really isn't a huge slowdown; *bind* isn't called *that* often a multi-megabyte PostScript stream. Also, most template comparisons are very quick, since they usually fail immediately upon comparison of the first items in the argument and template procedures.

I don't have occasion to use this mechanism very often, but when I do, it's a life saver.

Acrobat 9 and Acrobat.com



Acrobat 9 is released and introduces a nice collection of new features: portfolios, improvements to the typewriter and other tools, a new set of wizards for creating forms, etc. All really good stuff and things I'll be using frequently.

However, there's one new feature that has me very enthusiastic and it's not really part of Acrobat at all: Acrobat.com.

Acrobat.com is a free on-line site by Adobe Systems that provides a variety of web-based services. Among these are Adobe® Buzzword®, a web-based word processor; ConnectNow, a live conferencing service, etc; Share, which lets you share files with other people.

For me, however, the thing that is most exciting about Acrobat.com is that you can use it to vastly simplify the on-line distribution and collection of PDF-based forms. Acrobat 9 will automatically

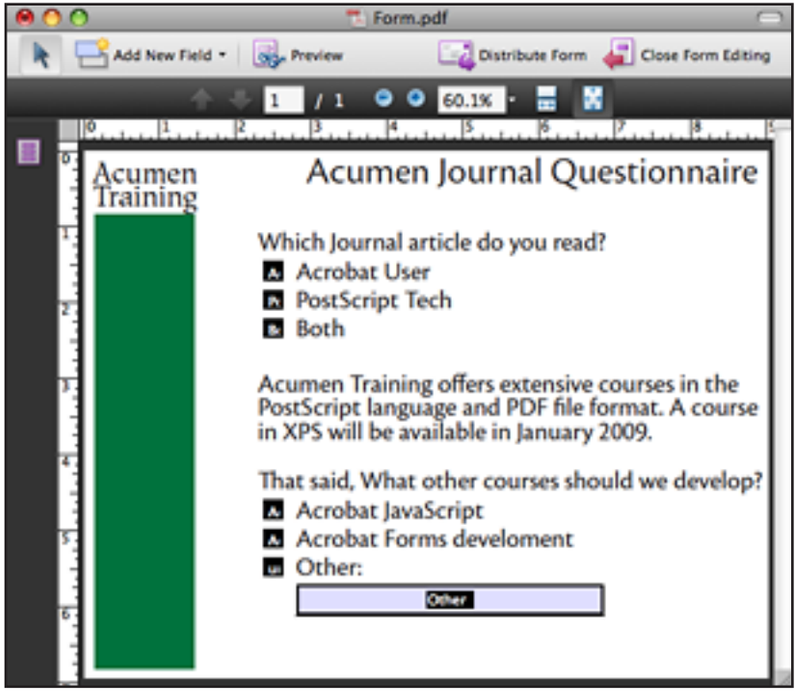
1. Post a PDF form onto Acrobat.com
2. Notify your list of users that the form is available, and then
3. Collect the responses back from those users by way of Acrobat.com.

It all works seamlessly, easily, and pretty much how you would want it to.

This is seriously cool stuff.

Creating Forms in Acrobat 9

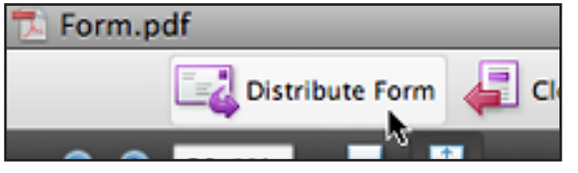
Although you can still place form fields on a PDF page using the Forms toolbar, the "9-native" way to create or modify forms is by entering a new "Edit Forms" mode, by selecting *Forms>Start Forms Wizard* (if you are starting a new form) or *Forms>Add or Edit Fields* (if you have an existing file or form open).



Either way, you will end up looking at your PDF page in Edit Forms mode, which is distinguished by having a set of buttons and a specialized toolbar at the top of the document window, as at right.

In this article, we won't discuss the details of placing form fields on the page using this window; we'll probably discuss this in a later issue, since many of the details have changed from previous versions of Acrobat.

Here, I want to call your attention to the *Distribute Form* button at the top of the window.

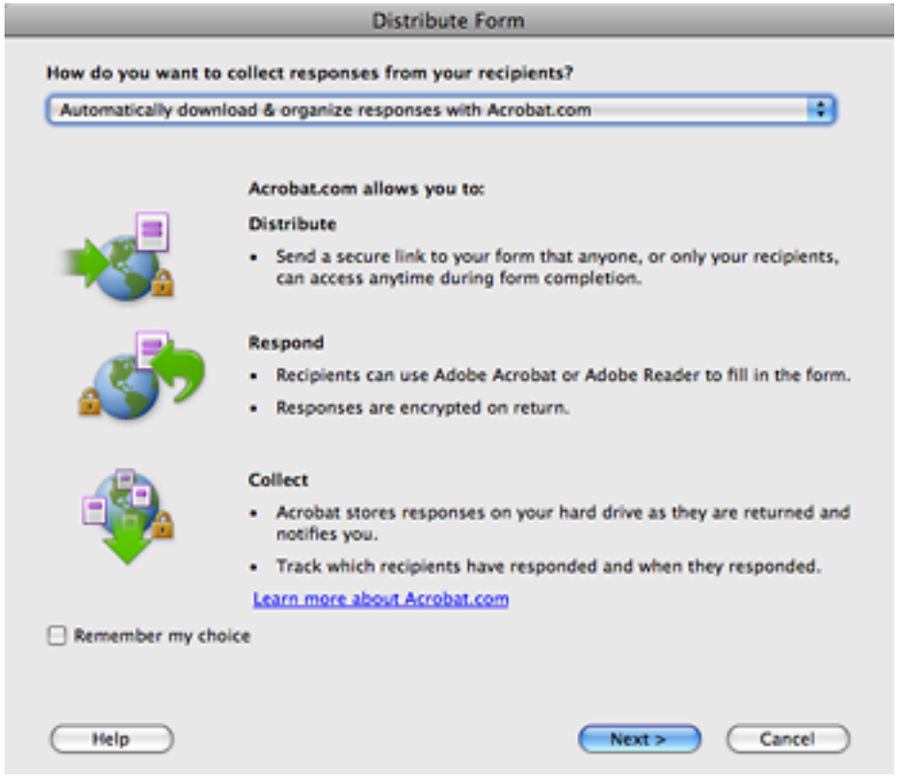


This is the entry point into a new way of approaching form distribution and collection. We are no longer going to email forms to people ourselves nor shall we necessarily post the forms on our own website (though we certainly can). Instead, we'll let Acrobat distribute the forms and collect the responses from our customers.

Having placed all the form fields on the page, you click on the Distribute Form button and here's what happens:

The *Distributing Forms* Wizard

Step 1. Choose a Distribution Method



Acrobat 9 uses a wizard interface to distribute your form. The first panel of this wizard asks how you want to distribute and retrieve responses from your form. You have three choices here, selectable in the pop-up menu:

- Distribute the form using Acrobat.com (this is our choice).
- Distribute the form by email.

  Acrobat will launch your web browser and email the form to a list of recipients.
- Distribute the form on your own server.

  You will need to make a directory on your server available to Acrobat for storing your form and all the ancillary files that Acrobat uses to track the form responses.

In our case, we're going to see how to use Acrobat.com for distribution and retrieval.

In the pop-up menu, we select *Automatically download & organize responses with Acrobat.com.* (Acrobat 9 consistently uses long, but unambiguous menu item names.)

When you click the *Next* button, Acrobat displays the next panel in the wizard.

2. Sign-in to Acrobat.com

The second pane in the Distribute Form wizard (right) asks you to sign-in to Acrobat.com.

Enter your Adobe ID (which is your email address) and password and click the *SignIn* button.

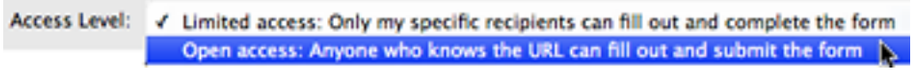
Note that there is a link here that lets you create a new Adobe ID if you don't already have one.

3. Assemble a Distribution List

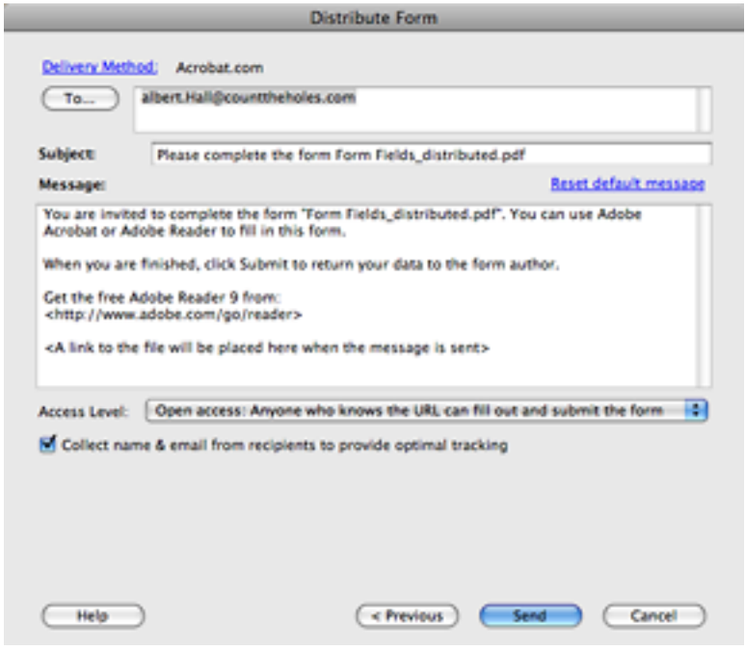
The next step in the wizard (below right) lets you create a list of people who should be notified of the form's existence. These people will receive an email with the URL of the form on Acrobat.com. All they will need to do to download and fill out the form is click on the link.

In this wizard panel, enter one or more email addresses into the *To* text box. The email addresses can be separated by spaces, commas, or semicolons.

You also must specify who is allowed to access the form: only the people whom you have notified or anyone who has the URL.



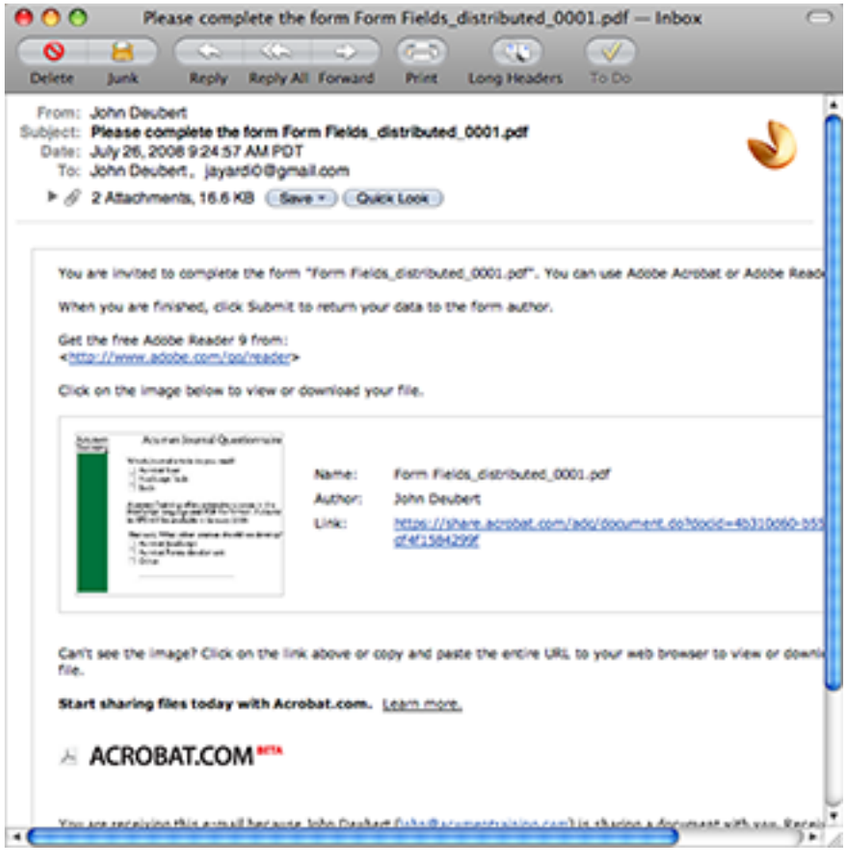
When you click the *Send* button, Acrobat will send the email to all the recipients you have listed.



Filling Out the Form

Retrieving the Form

When you receive an emailed notice that a form is ready for you to fill out, the message you receive will look something like the one at right.



This email message has two items of note:

- A thumbnail preview of the form document.
- A link to the form, stored on Acrobat.com.

When you click on the link, it launches your web browser, taking you to Acrobat.com, which informs you that the form can't be filled out online, but must be downloaded and opened locally; it provides you with a convenient button for doing this, as below.

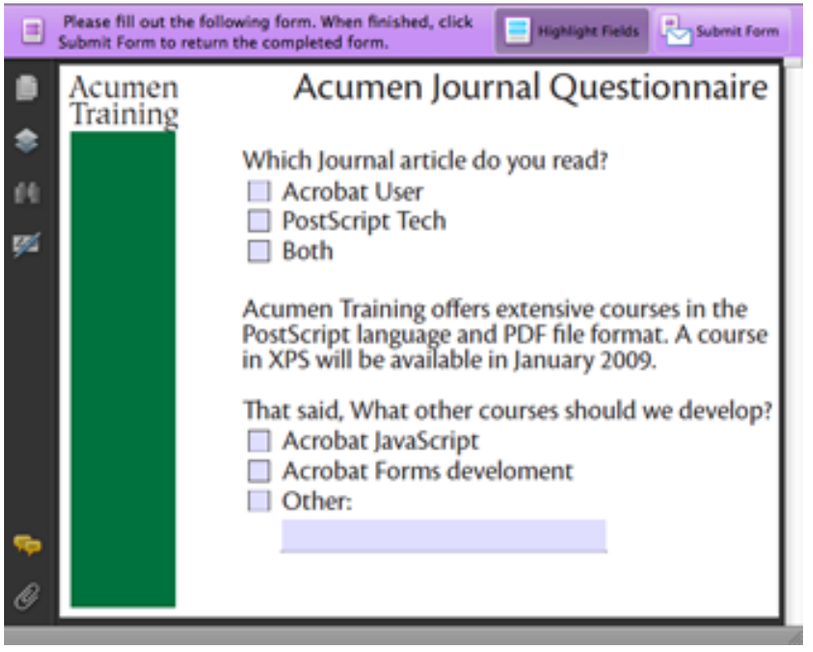
When you click on the *Download* button, Acrobat downloads the file to your hard disk and then opens it, so that you can fill out the form.

Try it!

I have placed this form on Acrobat.com for your downloading pleasure; click here to download it. You can also find it on the Acumen Training Resources page; look for *AcrobatDotCom.pdf*.

Filling out the form

When Acrobat 9 opens a form retrieved from Acrobat.com, it adds three items to the top of the document window:



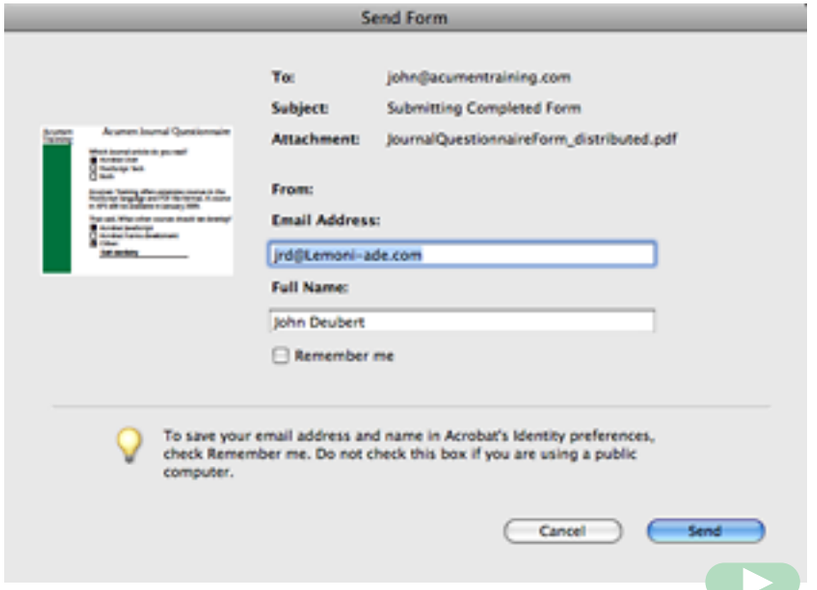
- Some cursory instructions telling you what to do with the form.
- A *Highlight Fields* button that toggles a light blue highlight that makes it easier to see where the fields are located on the document page. (This highlighting is visible at right.)
- A *Submit Form* button, which is the subject or our discussion here.

  Note that you no longer need to create your own Submit button as part of your form design; Acrobat gives you one for free.

You fill out the form they way you fill out any HTML- or PDF-based form, using the usual checkboxes, radio buttons, text fields, etc.

When you are finished, you click the *Submit Form* button.

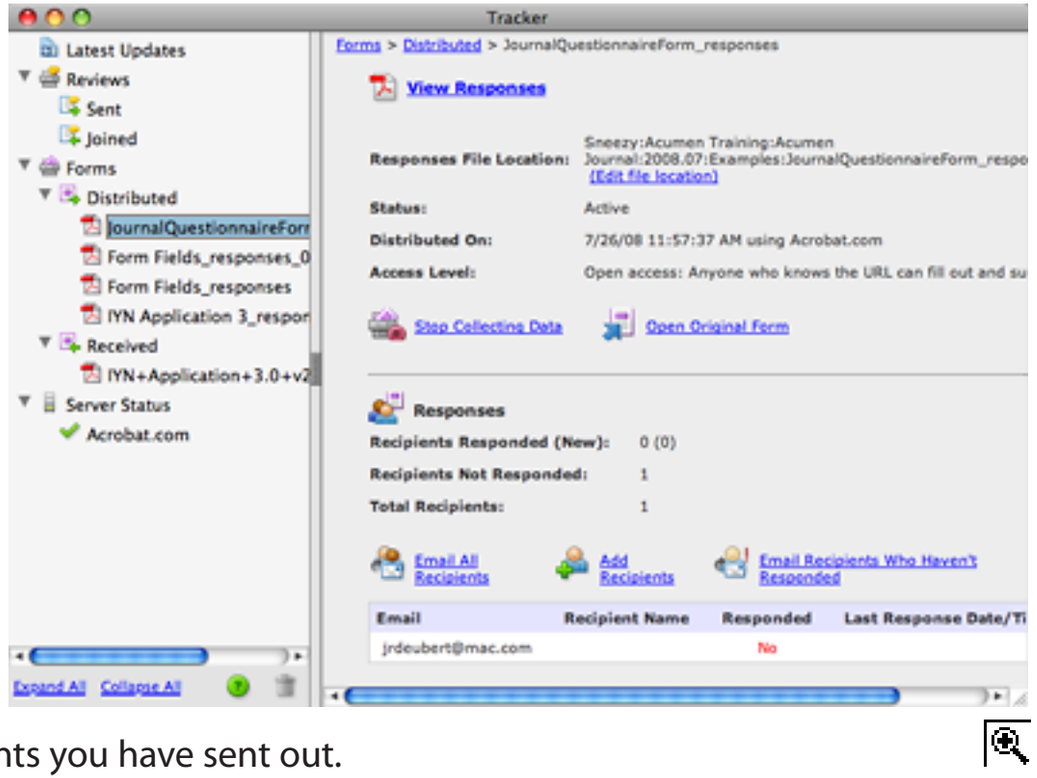
Acrobat presents you with the *Send Form* dialog box, which asks for your name and email address (so that the person who distributed the form can identify who it came from). Supply this information (if you wish) and click the *Send* button.



Acrobat sends the form data back to Acrobat.com.

Retrieving Results

Acrobat makes it very easy to retrieve the responses for a form you have distributed with Acrobat.com. The key is a tool called the *Tracker*. This Acrobat tool lets you track responses from forms you have distributed and documents you have sent out for review. It can track the responses whether you distributed them by email or by Acrobat.com, though in this article, we shall assume the latter.



You open the Tracker by selecting *Forms>Track Forms*. The resulting Tracker window displays two panels:

- The left side displays a hierarchical list of all the forms and reviewed documents you have sent out.
- The right side displays information about whatever list or review you select in the left-hand panel.

When you select a form in the list, the right-hand panel becomes populated with information about that form. In particular, at the bottom right of the window is a list of all the people to whom you have distributed the form and an indication of whether or not they have responded yet.

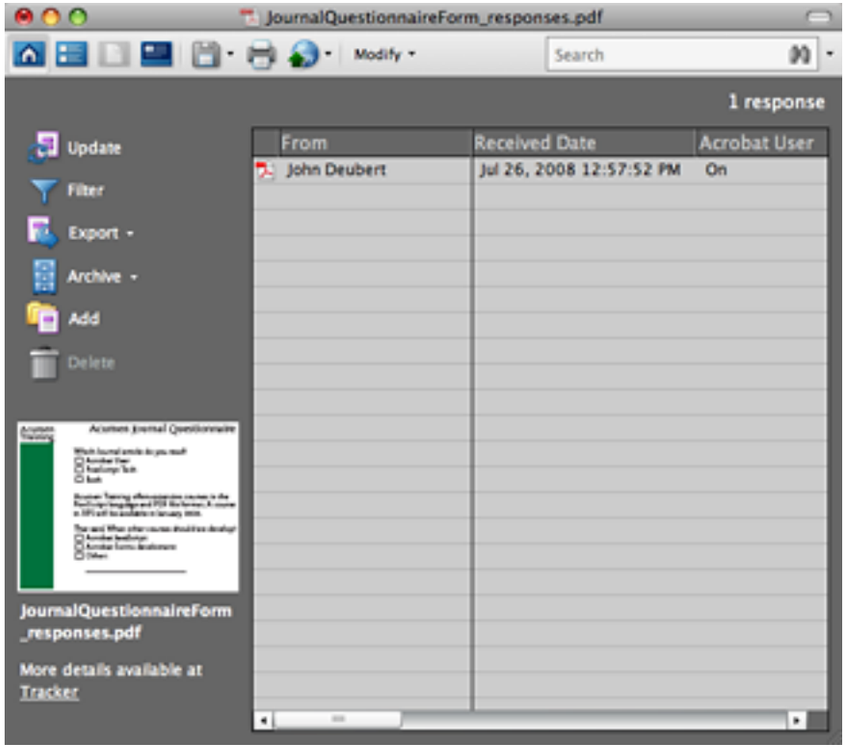
This will initially be "No", as at right.

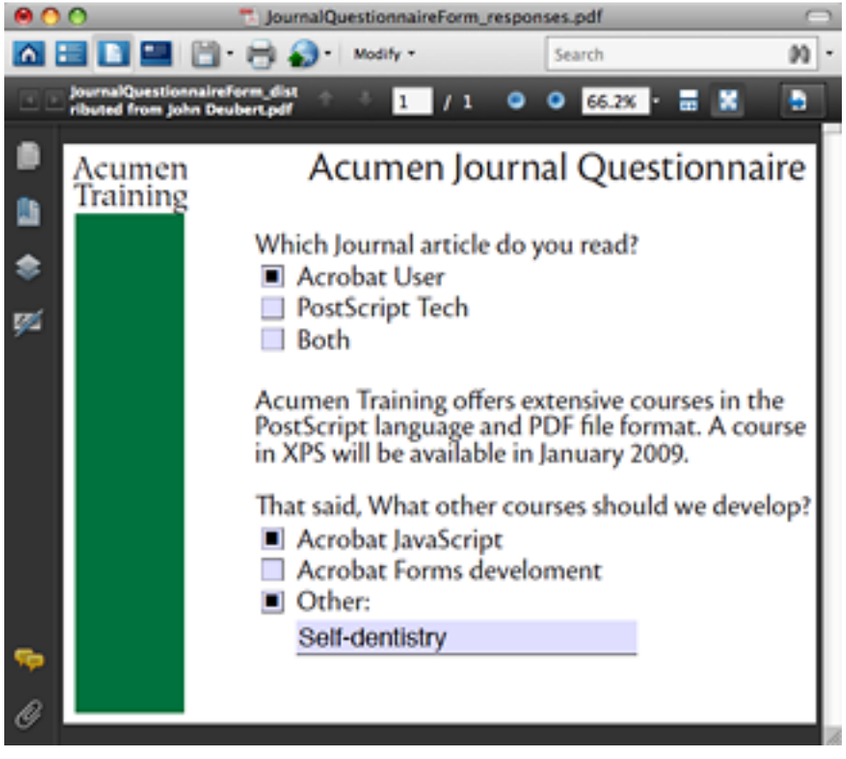
Email	Recipient Name	Responded	Last Response Date
jrdeubert@mac.com		No	

To collect more responses, click the *View Responses* link at the top right of the window. Acrobat will query Acrobat.com, retrieving all the new responses associated with this form and displaying them in a responses window (next page).

Viewing Form Returns

The Responses window (right) lists all of the recipients who have sent back responses to the selected form. If you double-click on one of the entries in the list, Acrobat displays the original form document with the fields all filled in with the respondent's answers.





Once you have examined the respondent's answers to your heart's content, close the document window to return to the Responses window so you can pick one of the other respondent's file.

Cool, Very Cool

I am very excited about Acrobat.com and its application to forms. It makes mindlessly simple something that was always a bit of a nuisance. You no longer need to have an IT organization on tap to arrange a server-moderated form distribution.

Schedule of Classes, August – October 2008

Following are the dates of Acumen Training's upcoming PostScript and PDF 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. See the Acumen Training web site for more information.

PDF Courses

PDF 1: File Content and Structure		Sept. 15–18	Oct 27–30
PDF 2: Advanced File Content			
Support Engineers' PDF	Aug 21–22		Oct 9–10

PostScript Courses

PostScript Foundations	Aug 25–29		Oct 13–17
Advanced PostScript		Sept 1–4	
Variable Data PostScript			Oct 20–24
Troubleshooting PostScript	Aug 18–20		Oct 6–8

Course Fee Classes cost $2,000 per student, except for *Troubleshooting PostScript* and *Support Engineers' PDF,* which are $1,500 per student. 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?

Acrobat 9 Visual Quickstart Guide

The Acrobat 9 Visual Quickstart Guide should be on the bookstore shelves within a week or two of your reading this. This book covers everything you need to know to use Acrobat 9 for PDF document creation, review, distribution, security, and more!

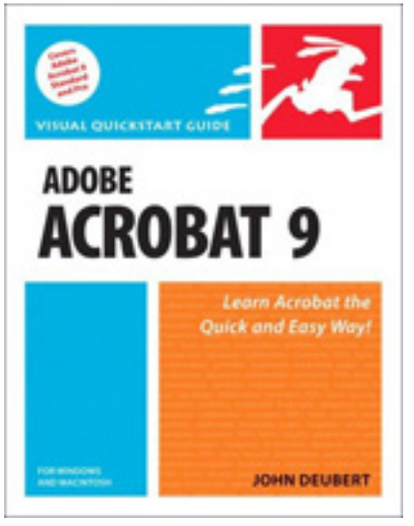


Table of Contents

The book discusses all the important features of Acrobat Standard and Pro.

Buy several!

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?

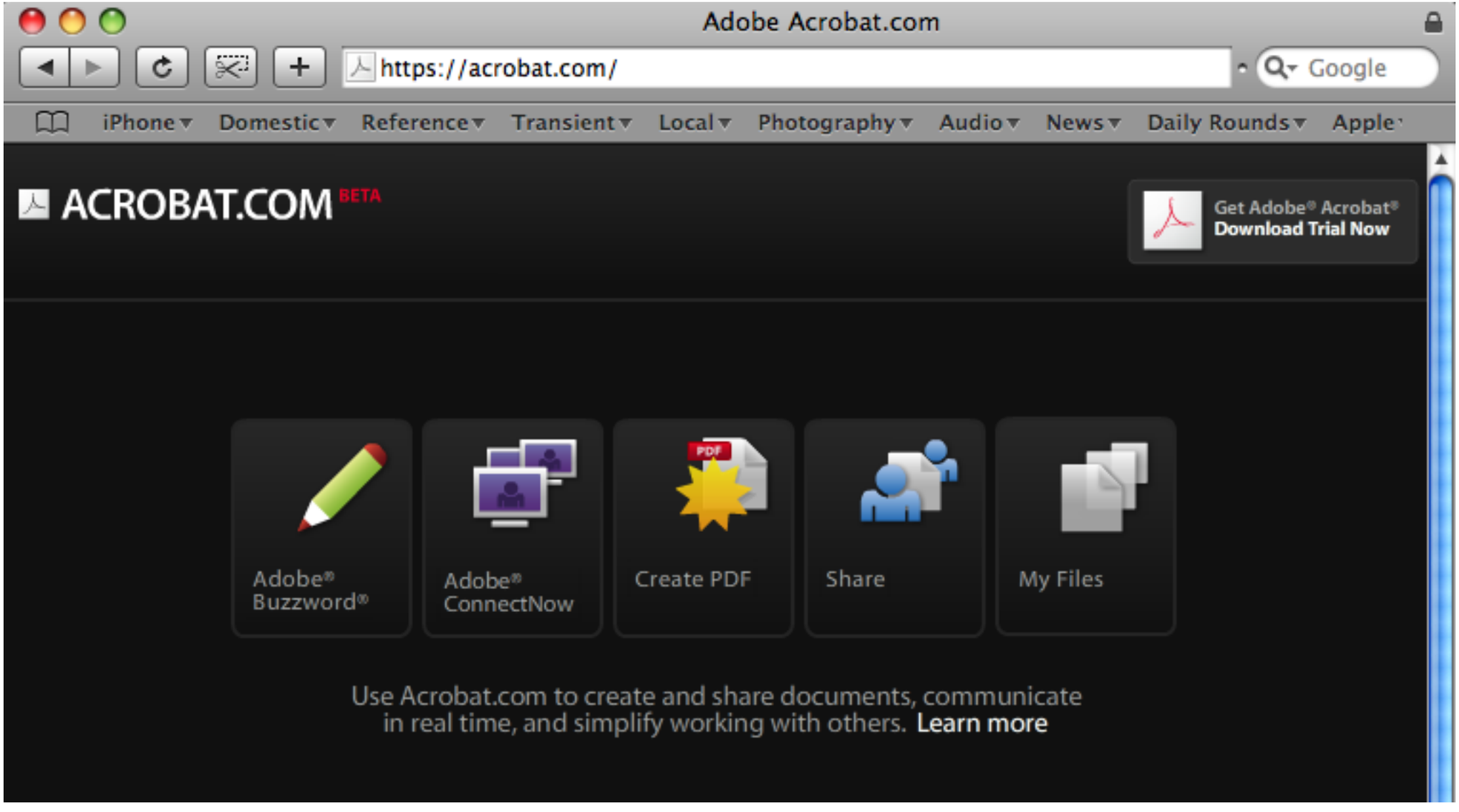
Suggestions for articles. Each Journal issue contains one article each on PostScript and Acrobat. What topics would you like me to write about?

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

Acrobat.com Home Page



Emailed Announcement

Please complete the form Form Fields_distributed_0001.pdf — Inbox

Delete Junk Reply Reply All Forward Print Long Headers To Do

From: John Deubert
Subject: **Please complete the form Form Fields_distributed_0001.pdf**
Date: July 26, 2008 9:24:57 AM PDT
To: John Deubert , jayardi0@gmail.com

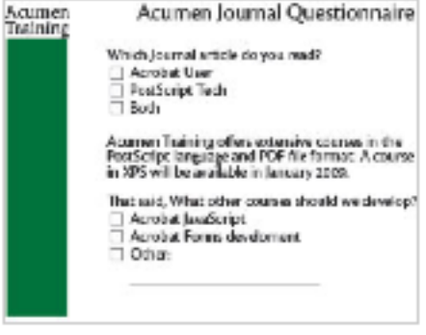
2 Attachments, 16.6 KB Save Quick Look

You are invited to complete the form "Form Fields_distributed_0001.pdf". You can use Adobe Acrobat or Adobe Reade

When you are finished, click Submit to return your data to the form author.

Get the free Adobe Reader 9 from:
<http://www.adobe.com/go/reader>

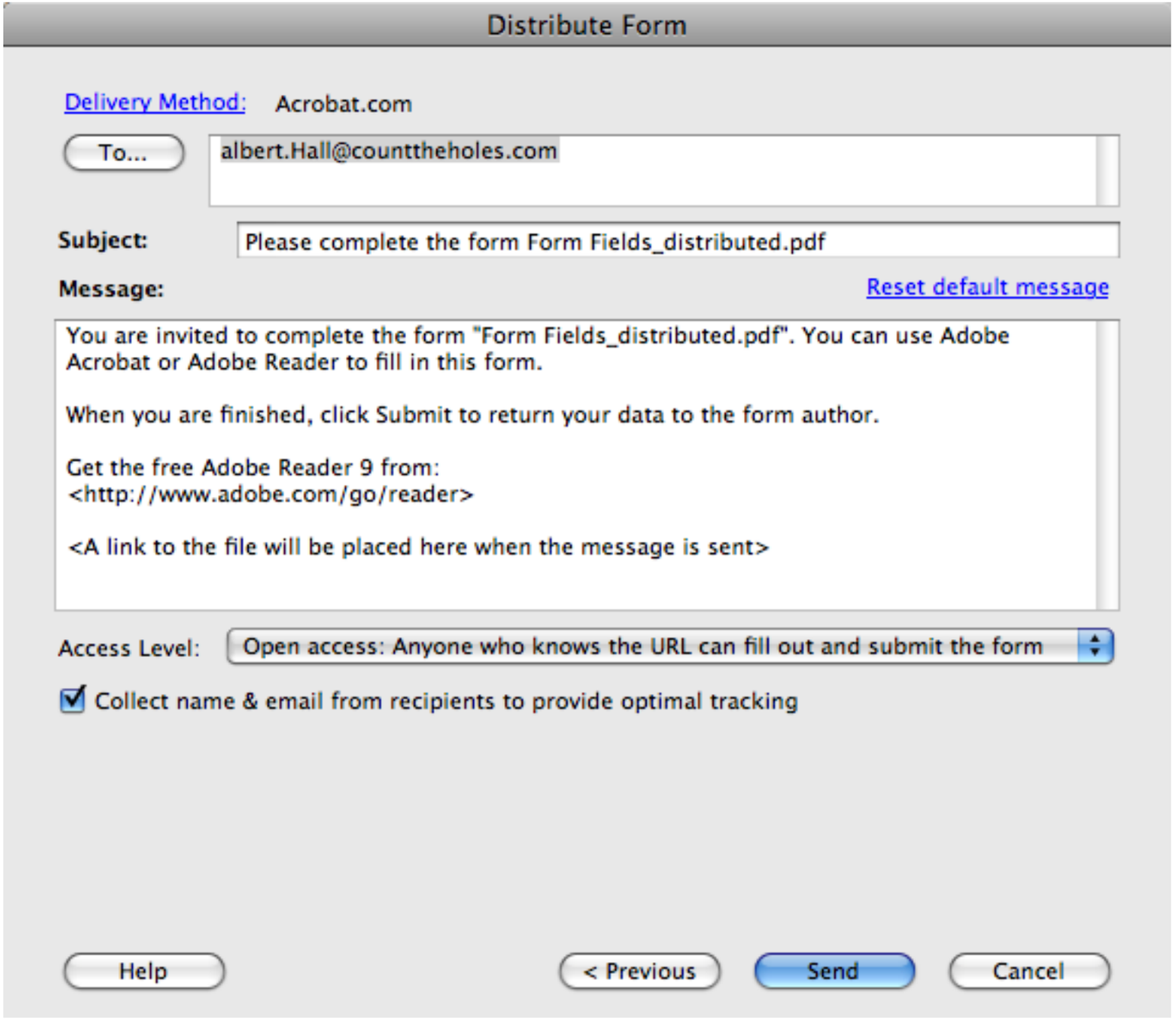
Click on the image below to view or download your file.



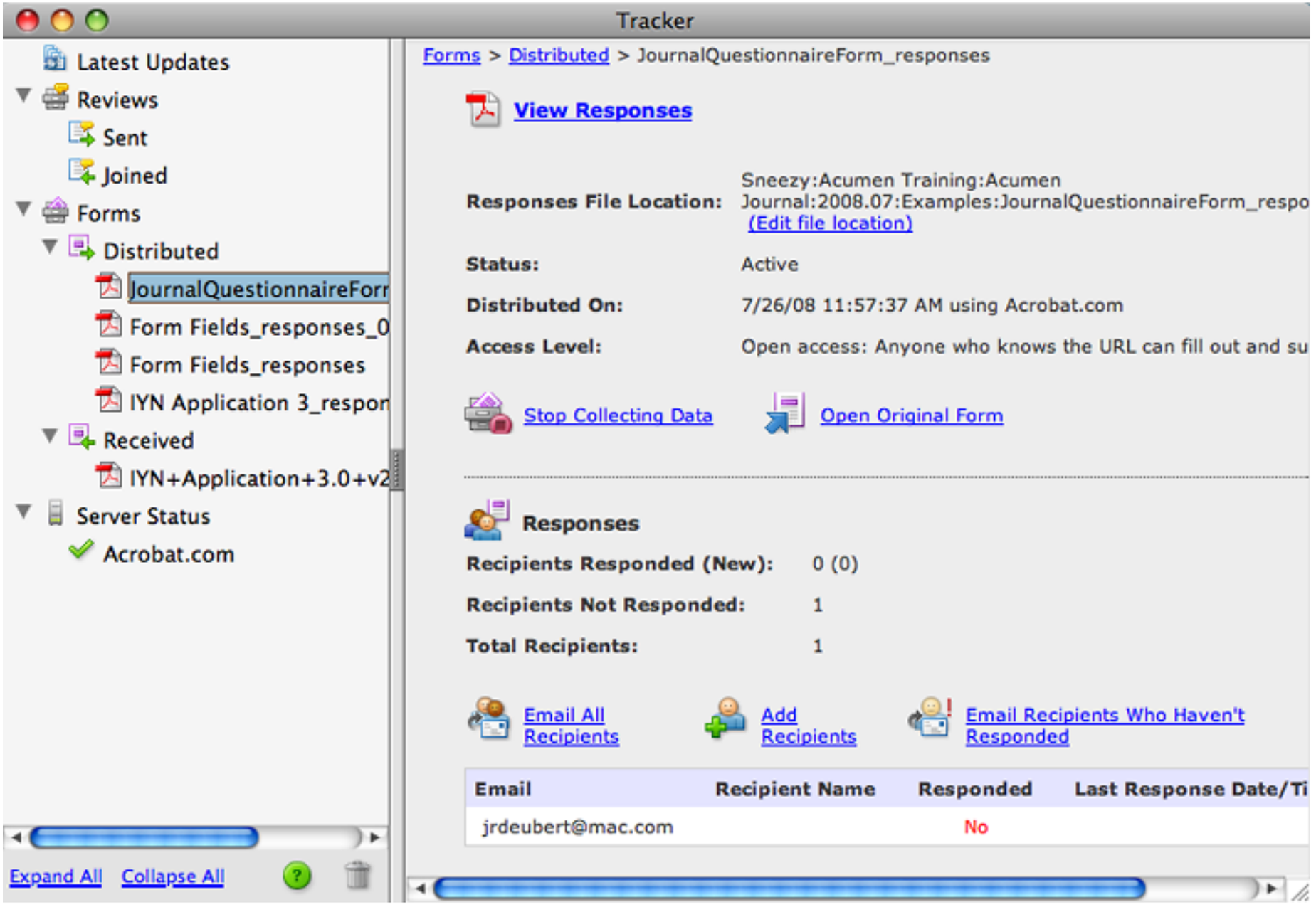
Name: Form Fields_distributed_0001.pdf
Author: John Deubert
Link: https://share.acrobat.com/adc/document.do?docid=4b310d60-b55
df4f1584299f

Can't see the image? Click on the link above or copy and paste the entire URL to your web browser to view or downl
file.

Distribute Form



Tracker Window



Form Responses Window