

# Adobe® Acrobat®

## Forms System Implementation Notes

---

### Introduction

Acrobat Forms is a group of extensions added to Portable Document Format (PDF) in version 1.2, for use with Acrobat software products. These extensions allow Acrobat users to create PDF forms that contain fields and buttons. Users of Acrobat or Acrobat Reader software can complete and return PDF forms via e-mail or the Web. PDF Forms features are simply a new layer of capabilities on top of a PDF file. The underlying PDF file may be created from any authoring application by any PDF producing tool within Acrobat software such as PDF Writer, the Acrobat Distiller® application, or Acrobat Capture® software. Form fields are subsequently added using Acrobat.

In addition to the forms data stored in PDF files, Acrobat products also support two formats for transmitting forms data: the HTML form format (MIME type **application/x-www-form-urlencoded**), which transmits data from the client to the server, and an Acrobat software-specific format called Forms Data Format, or FDF (MIME type **application/vnd.fdf**), which transmits data between the client and the server.

This document provides an overview of the various architectures for Acrobat Forms applications. It also compares HTML and FDF file formats for Acrobat Forms applications and helps you decide when to use each one.

### Prerequisites

To get the most from this document, you should be familiar with the following:

- The process of creating form fields and buttons, and modifying their properties. The Acrobat 4.0 online documentation covers these—in particular, the Forms Online Guide, accessible via the menu item Help > Acrobat Guide.
- The process flow for HTML forms, on both the client and the server. Technical bookstores generally stock many excellent books that cover this topic.
- To take full advantage of the capabilities of Acrobat Forms, you should be familiar with FDF. FDF is described in an appendix of the Portable Document Format Reference Manual, which is available at [www.adobe.com/prodindex/acrobat/pubs.html](http://www.adobe.com/prodindex/acrobat/pubs.html).

### Other sources of information

There is an Acrobat Forms Resources page at [www.adobe.com/prodindex/acrobat/acrforms.html#formsres](http://www.adobe.com/prodindex/acrobat/acrforms.html#formsres).

The FDF tool kit page is located at <http://beta1.adobe.com/ada/acrosdk/forms.html>.

### The basics

When users fill out an Acrobat form, they must click a button to submit the data from the form to the server. This is called a **submit form** action. The format of the submitted data can be either HTML-compatible (urlencoded) or FDF. This decision is made when the form is created. In the same dialog box in which you, as the form's creator, enter the URL to which the data is submitted you also specify which form format to use to transmit the data.

If you select HTML, the format is identical to and compatible with any existing HTML form. Existing Common Gateway Interface (CGI) scripts for HTML forms may be used to parse data in this format.

If you select FDF, there is a server library to help parse and generate FDF files. The URL you submit to is not restricted to the http scheme. It can also be the **mailto** scheme, for example, **mailto:someuser@somecompany.com**.

**Note:** To see the format of the FDF data that is being sent to the server, create a form and enter data into one or more of its fields. Instead of submitting this to the server, simply select the Export Form Data menu item from the file menu of Acrobat Exchange.

## Choosing the output format

This section describes how to choose between HTML and FDF formats when implementing an Acrobat Forms application.

HTML support allows Acrobat forms to be used as a direct replacement for HTML forms. Choose HTML if you need compatibility with existing systems.

FDF supports various capabilities that HTML does not. Choose FDF if you wish to take advantage of its additional features, which include the following:

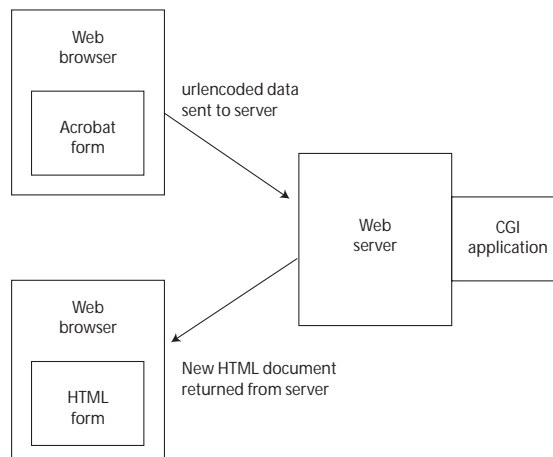
- When sending data back from the server, you do not need to re-send the form itself; the data can populate the same form that originated the data.
- You can alter the appearance of the buttons. You can also take advantage of this capability to send graphical information in either direction between the client and the server.
- You can alter the form by sending an FDF file from the server. You can re-program various actions attached to buttons (including adding new JavaScripts, changing the URL for a submit form action, etc.). You can also change field properties, such as hidden, read-only, required, and don't print. And, you can populate list boxes and combo boxes with different choices, etc.
- You can use FDF to dynamically synthesize PDF documents composed of a variable number of pages, from templates found in PDF documents specified by the FDF data. You can also use FDF to populate any fields in the **spawned** pages with data. A template is simply a page with a name attached and may be visible in its source PDF document or hidden.

## Replacing an HTML form with an Acrobat form

**Figure 1** shows the simplest case, which permits existing CGI applications to be used without modification. To use Acrobat Forms in such a system, you simply create:

1. An Acrobat form with field names that match those in the existing HTML form.
2. A button on the form whose action is a submit form action. The URL to submit to may be relative to the URL of the form that you are submitting from.

Figure 1: Using HTML with Acrobat Forms



### Acrobat form with results returned in the same form

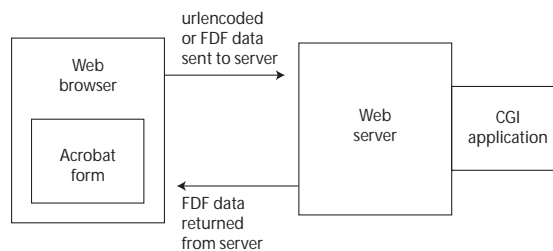
**Figure 2** shows a system in which the form data is returned into the same form as that from which it was submitted. In such a system, the data sent to the server may be either FDF or urlencoded format, while the data returned from the server must be in FDF format and must have a MIME type of **application/vnd.fdf**.

**Note:** When the server returns data in FDF, the URL to submit to *must* end in #FDF. For example, <http://yourserver.com/cgi-bin/yourscript#FDF>

Existing CGI applications must be modified to return FDF instead of HTML documents. **Example 1** shows a simple FDF-generating Active Server Page (ASP) that works with the Microsoft® Internet Information Server 3.0. For anything more complicated than this example, the use of the FDF tool kit is highly recommended. See “Other sources of information” on page 1.

**Note:** If you return a static FDF file stored on the server, as opposed to one dynamically generated by a server script as in the example below, then you may have to define application/vnd.fdf as a new MIME type on your server.

Figure 2: Returning FDF into the same form



### Example 1: Simple ASP for generating FDF data

```

<%@ LANGUAGE = VBScript%>
<% Response.ContentType = "application/vnd.fdf" %> %FDF-1.2
1 0 obj
<<
/FDF << /fields [
<< /T (status)/V (Hello, World!) >>
] >>
>>
endobj
trailer
<<
/Root 1 0 R
>>
%%EOF
  
```

### Acrobat form with results returned in a new form

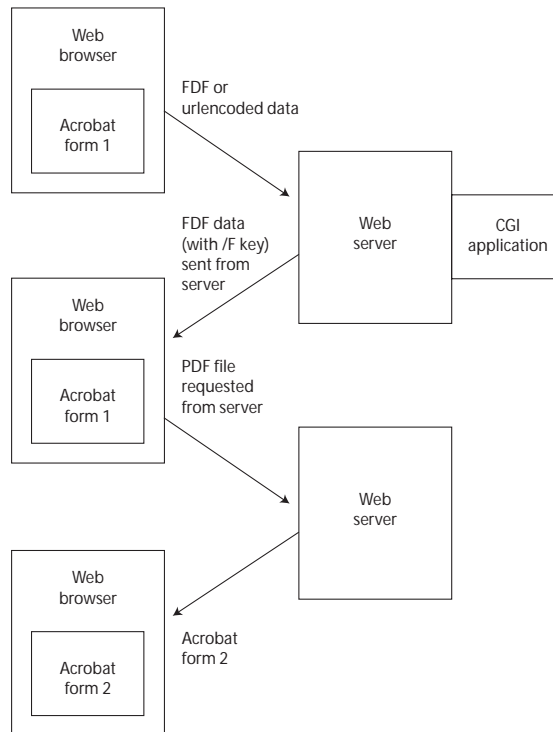
In some cases, you may wish to return data in a different form, not the form from which it was submitted.

**Figure 3** shows such a case. The modified CGI application needed to implement such an application is almost identical to that needed for the system described in the previous section. The only difference is that you must include an /F key in the FDF file when you want to populate a different form. The value of the /F key specifies the URL for the PDF file to populate with the forms data. This URL may be relative to the URL of the form that you are submitting from. The specified file is retrieved (from the server) and populated with the forms data.

**Note:** If the returned FDF data is for the same form that you submitted from, it should not include the /F key.

The data sent to the server may be either FDF or urlencoded. The data returned from the server must be in FDF and have a MIME type of **application/vnd.fdf**.

Figure 3: Returning FDF into a different form



#### HTML form with results returned in an Acrobat form

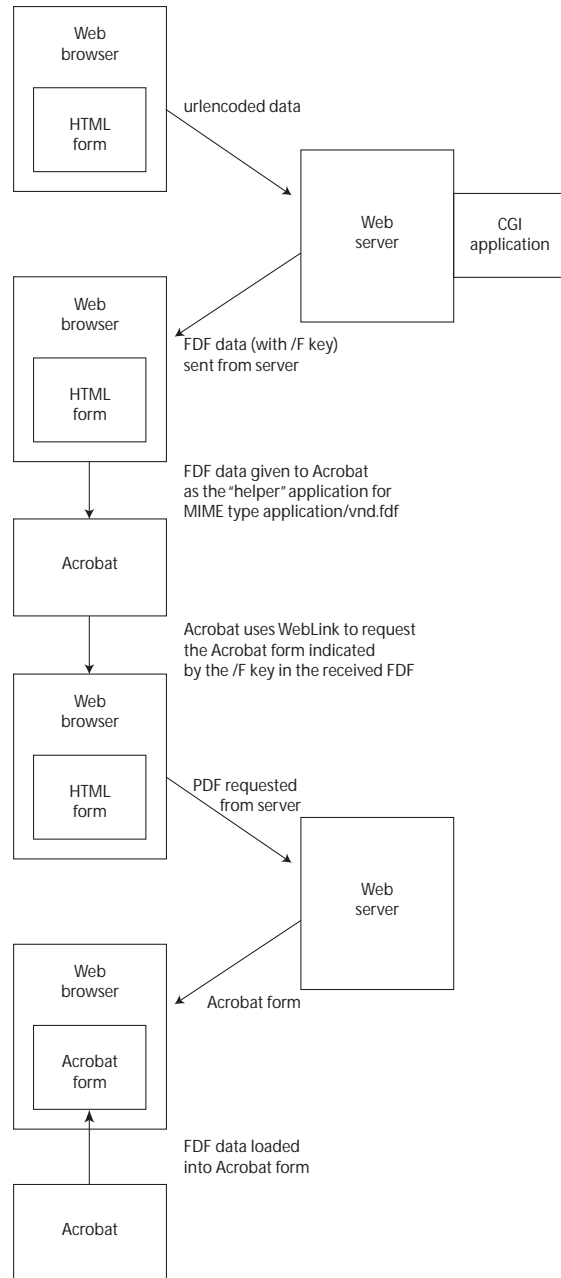
An additional possibility is starting from an HTML form, submitting to the server, and having the server return an FDF file whose /F key gives as value the *absolute* URL of an Acrobat form. The specified form is retrieved (from the server) and populated with the Acrobat Forms data. **Figure 4** shows such a system.

**Note:** For this to work, you must select Acrobat as the application that handles the MIME type **application/vnd.fdf**. For example, if you are using Netscape (e.g., Communicator 4.x), then you do this under Preferences > Applications. Choose Acrobat as the “helper” application. If you are using Internet Explorer on the Microsoft Windows platform, open Windows Explorer, choose the menu item View > Options > file Types, and make sure there is an entry for Adobe Acrobat Forms Document that has the Default Extension for Content Type: set to FDF and the Content Type (MIME): set to “application/vnd.fdf.” Additionally, the checkbox “Confirm open after download” should be unchecked. Finally, under “Actions:” there should be an entry for “open,” and in its properties, “Application used to perform action” should be set to C:\Program Files\Adobe\Acrobat 4.0\Acrobat\Acrobat.exe (or wherever Acrobat resides), and the checkbox “Use DDE” should be unchecked. It also is imperative that from within Acrobat you go to the file > Preferences > Weblink menu item and choose your browser. (Note that if you are using Acrobat 4.0, these configurations were likely made for you when you installed Acrobat 4.0.)

**Note:** To return FDF data from an HTML form, the URL to submit to does not need to end in #FDF, unlike the cases described earlier in “Acrobat form with results returned in the same form” on page 3 and “Acrobat form with results returned in a new form” on page 3.

**Note:** In Microsoft Internet Explorer, the PDF file will open in a new browser window from the one displaying the HTML that the submission was triggered from (this has been fixed in Acrobat 4.0). In Netscape Communicator 4.x, the PDF file opens in the same window, and even in the same frame (if frames are being used).

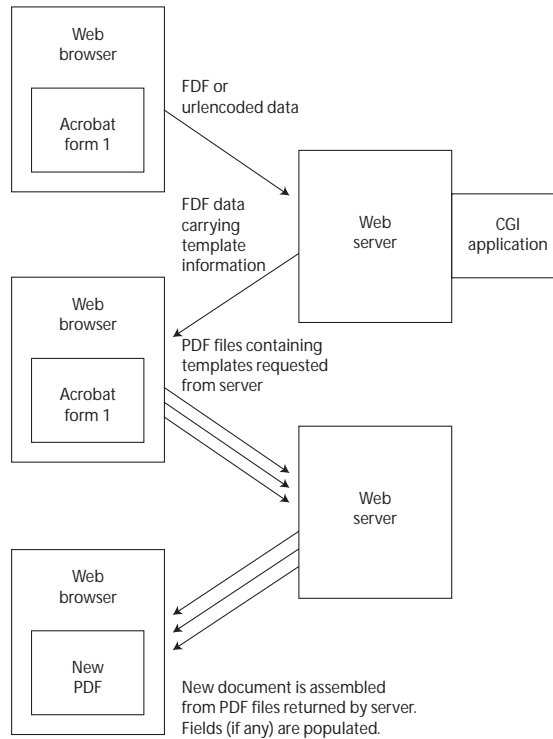
Figure 4: Start from HTML, end in an Acrobat form



## Templates

A recent addition to Acrobat Forms is the ability to dynamically create PDF documents composed of a variable number of pages, from templates found in PDF documents specified by the FDF, and to populate any fields in the “spawned” pages with data carried by that FDF. Figure 5 shows such a system.

Figure 5: Dynamic creation of a PDF file from templates



**Adobe Systems Incorporated**  
345 Park Avenue  
San Jose, CA 95110-2704 USA

**Adobe Systems Pty. Ltd.**  
Level 4, 67 Albert Avenue  
Chatswood, NSW 2067  
Australia

**Adobe Systems Europe Limited**  
Adobe House, Mid New Cullins  
Edinburgh EH11 4DU  
Scotland, United Kingdom

**Adobe Systems Co., Ltd.**  
Yebisu Garden Place Tower  
4-20-3 Ebisu, Shibuya-ku  
Tokyo 150-6017 Japan

**World Wide Web**  
[www.adobe.com](http://www.adobe.com)

This brochure was created with Adobe PageMaker® software and font software from the Adobe Type Library.

Adobe, the Adobe logo, Acrobat, Acrobat Capture, Acrobat Exchange, Distiller, and PageMaker are trademarks of Adobe Systems Incorporated. Microsoft and Windows are either trademarks or registered trademarks of the Microsoft Corporation in the United States and other countries. Macintosh is a trademark of Apple Computer, Inc. registered in the United States and other countries. Netscape and Netscape Communicator are trademarks of Netscape Communications Corporation. All other trademarks of the property of their respective owners.

© 1999 Adobe Systems Incorporated. All rights reserved. Printed in the USA. BC1317 1/99

The information in this document is furnished for informational use only is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies that may appear in this document. The software described in this document is furnished under license and may only be used or copied in accordance with the terms of such license.

Version History: Original version April 8, 1997. Updated April 30, 1998