PDF documents
FDF forms
HTML

Use FDF Forms with Domino Instead of HTML Forms

By John Duggan

Do HTML forms give you enough control or consistency in how they look? Does your latest masterpiece look wonderful under Windows, but terrible under UNIX, and even worse printed?

Instead of tearing out the remaining hair left on your head, try using Forms Data Format (FDF) forms. They're easy to use and remarkably straightforward with Domino. FDF forms are powerful and are a good replacement for HTML forms because they provide a precise layout, consistent printing, and look identical to their paper counterparts.

Users must have at least Acrobat Reader 3.0 and the latest FDF toolkit must be installed on the Domino server.

This article shows how to send an FDF form from a Web client to a Domino server and send an FDF form from a Domino server to a Web client.

PDF documents

The major selling point of Portable Document Format (PDF) documents is they display and print how the document was created. It does not matter what fonts, software, or operating system is installed, the result is the same when the document is printed or viewed on screen. HTML forms are the opposite; they often differ across Web browsers and different operating systems. There is no guarantee how an HTML form will look to users (figure 1).

Acrobat Reader reads PDF documents, is readily available, and is free. Adobe Acrobat is a tool that allows conversion of electronic files into PDF files. There are a variety of third-party tools that can perform the conversion, but there are very few that can create and use interactive FDF forms.

Subscriber Download

- Example of how to use an interactive PDF document with Domino
- Filename: DUGGJ01.zip

FDF forms

FDF forms are an extension to PDF. The extension lets PDF

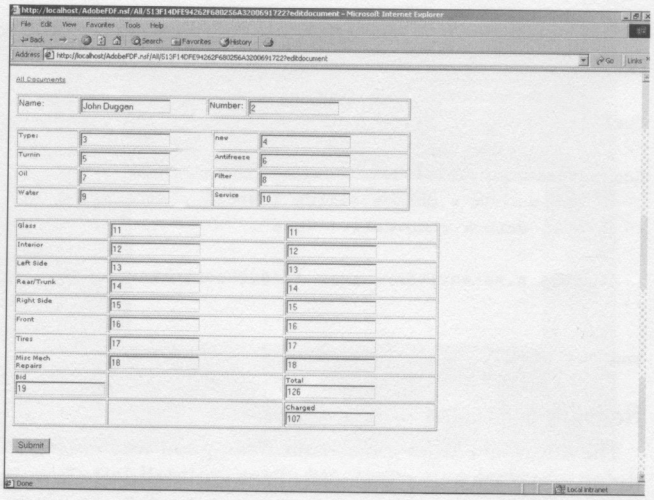

***Figure 1:* Standard HTML form**—This may look different to different users.

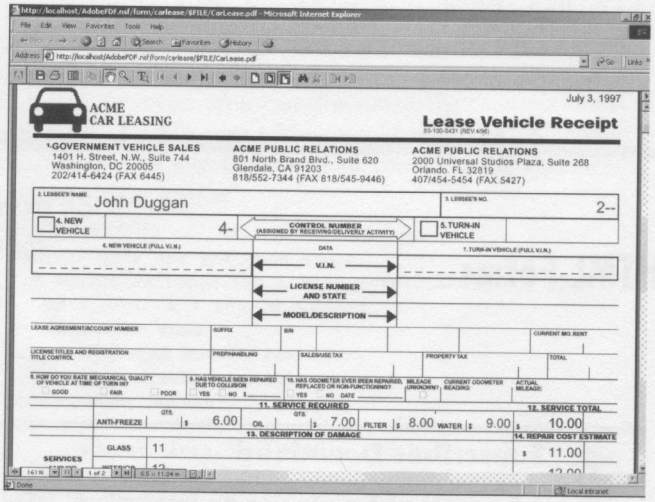

***Figure 2:* PDF document**—Here's the same HTML form as a PDF document with some fields filled.

documents contain form objects similar to HTML form objects including buttons, check boxes, combo boxes, list boxes, radio buttons, text boxes, and JavaScript.

FDF form objects also include functionality for mandatory fields, validation, calculations, actions and a range of useful hiding attributes.

In a nutshell, FDF forms is a file that Acrobat Reader processes and layers the result on top of a PDF document. Here's an example of an FDF file:

```
%FDF-1.2
1 0 obj
<< /FDF
<< /Fields
[
<< /V (John Duggan)/T (Name)>>
]
/F (CarLease.pdf)
>>
>>
endobj
trailer
<</Root 1 0 R >>
%%EOF
```

You can write a library to create the file manually from LotusScript, but there is a free toolkit from Adobe that parses and/or generates this file.

The FDF Toolkit is available for Windows NT or UNIX, and supports a range of development languages such as C/C++, Visual Basic, Java and Perl.

Installing the FDF toolkit on Domino

Here are the steps to use the FDF toolkit on a Domino server:

1. Download and install the FDF toolkit from the Adobe Web site.
2. Copy the files FDFAcX.dll and FdfTk.dll from the toolkit to the system directory \winnt\system32. (or a directory that has "execute" permissions and is within the system path).
3. Register the DLL—FdfAcX.dll. This DLL is an ActiveX component that must be registered with Windows. Type the following at the command prompt within the system directory: regsvr32 FdfAcX.dll

The FDF toolkit is now installed on the server and can be accessed from LotusScript by initializing a new object of the type "FdfApp.FdfApp" as shown below.

```
Set FdfAcX = CreateObject("FdfApp.FdfApp")
```

You can find the full list of methods for this object within the FDF reference documentation located in the FDF toolkit.

Send an Adobe form to a Web user

The first step is to create your own PDF document. This could be an existing paper document, Microsoft Word document, picture, or even a Web page. Acrobat generates PDF documents from a range of sources.

Next, add the form objects to the PDF document using Acrobat (figure 3).

The only requirement for using the PDF document on the Web is to provide a button whose action is a submit form

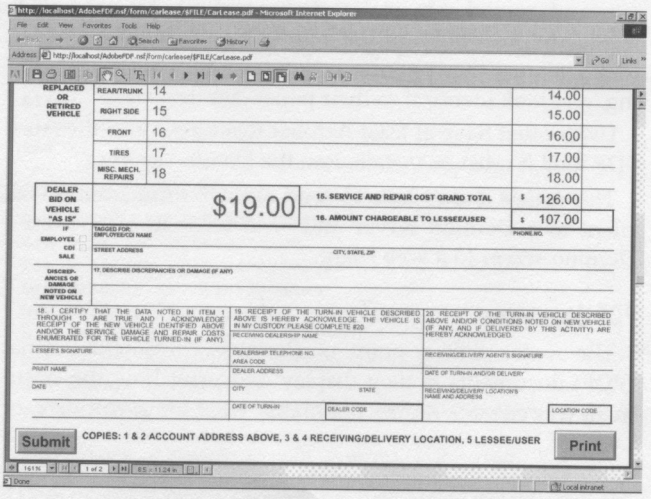

***Figure 3:* Acrobat**—Adding form objects to a PDF document

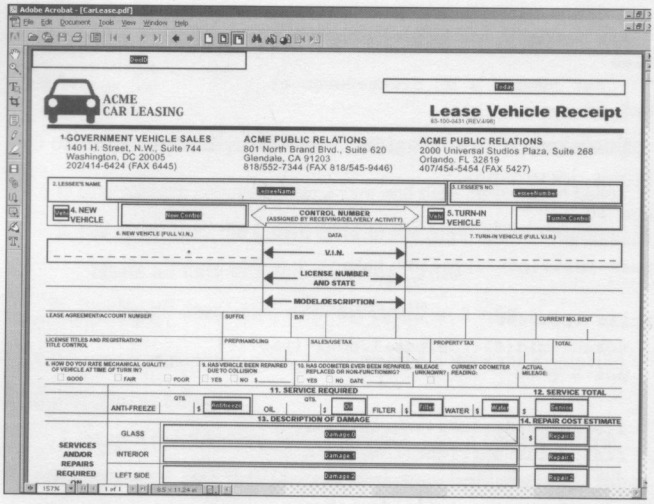

***Figure 4:* Buttons on the form**—The hide properties have been set on the buttons to disappear when the document is printed.

which contains the URL of where to send the form data (figure 4).

The final step is to provide server code that generates the FDF file. Within the example database, I have implemented a Notes agent called "Generate FDF" that creates an FDF file from a Notes document. The agent requires the unique document ID as a parameter.

Web agents can receive a parameter by inserting data after the & symbol within the URL.

GenerateFDF?OpenAgent&513F14DFE94262F680256A3200691722

Extracting the Document Unique ID from the above URL within the agent is straightforward by reading the contents of the CGI variable Query_String:

```
Set contextdoc = session.DocumentContext
id = Straight(contextdoc.Query_String(0),"&")
```

FDF forms require the Content-Type header to be replaced with application/vnd.fdf. This tells the Web browser the data it received is an FDF file. A print statement at the beginning of the agent tells Domino not to provide its own Content-Type:

```
Print "Content-Type:application/vnd.fdf"
```

If there are any problems with the FDF, send the data as text and you can read the file within the Web browser.

```
Print "Content-Type:text/html"
```

Initialize the FDF toolkit with the creation of a new FDF file.

```
Set FdfAcx = CreateObject("FdfApp.FdfApp")
Set FDF = FdfAcx.FDFCreate()
```

Because the agent is generic, put all the Notes fields into the FDF document except the system fields. It doesn't matter if there are too many fields, as the additional fields don't show within Acrobat Reader.

```
Forall i In opendoc.Items
    FieldName = i.Name
    Value = i.Values(0)
    If Left(FieldName,1) <> "$" Then _
        Call FDF.FDFSetValue (FieldName, Value, False)
End Forall
```

Tell the FDF file which PDF document to open. This can be a file on a network drive or a URL pointing to the PDF document. The PDF document in the example database is located within the same database. Set this value by calling FDFSetFile.

```
Call FDF.FDFSetFile ("http://" &
contextdoc.Server_Name(0) & _"/" & db.filename &
"/form/carlease/$FILE/CarLease.pdf")
```

After filling in an Acrobat form, the user must click on a button whose action is a submit form action to submit the data to the server (figure 5).

The action contains a URL that can either be hard coded within the PDF document or specified within the FDF file. The setting in the FDF file overrides the PDF document's setting. The URL is not restricted to http as mailto can be used if the data is sent by e-mail. Set this value by calling FDFSetSubmitFormAction.

```
Call FDF.FDFSetSubmitFormAction ("Submit", 3, _
"http://" & doc.Server_Name(0) & "/" & db.filename &
"/ProcessDocument?OpenAgent",5)
```

The FDF file is now complete and can be sent to the user. There is a function called FDFSaveToStr that passes the whole file into a string variable. Print the string and the file is sent to the Web user.

```
filestuff = FDF.FDFSaveToStr()
Print filestuff
```

The Web user has now received an FDF file, which launches Adobe Forms plug-in, loads the PDF document from the specified location and inserts all the fields on top of the PDF document.

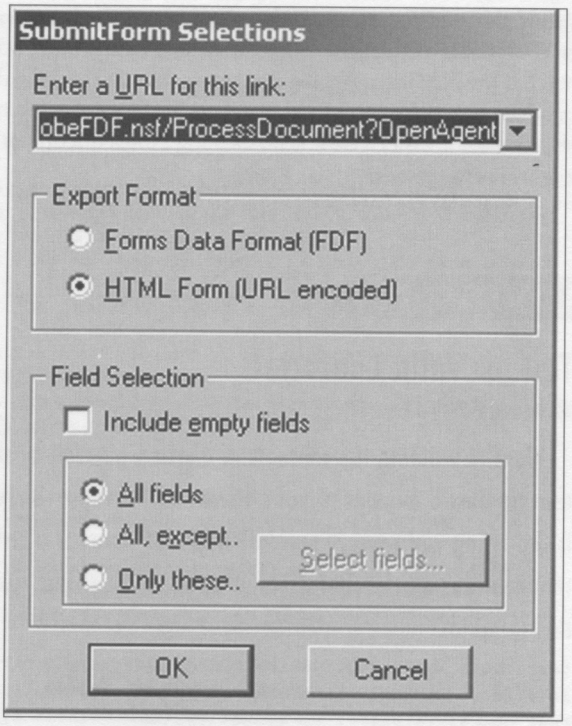

***Figure 5:* SubmitForm action**—These are the options available.

Submit an Adobe form to a Domino server

After a user fills in the fields and clicks on Submit, how do you read the form data from an Adobe PDF document?

The submitted form data can be sent as HTML or as FDF. The selection of which format to use is specified when the Submit Action button is created. The property is located within the same dialog in which the URL is specified as shown in figure 5. Acrobat version 5 has an additional option of sending the form data as XML.

There are advantages to submitting the data as FDF:

- When sending data back from the agent, the form does not have to be resent; the data can populate the original form.
- It can change the appearance of the form such as the buttons, field properties, populating list boxes, or combo boxes.

Parsing FDF data requires the toolkit whereas the HTML option doesn't and can process the data within Lotusscript. For the purposes of this article, assume the form data is sent as HTML.

In the example database, the Adobe form is submitted as HTML to an agent called "ProcessDocument." This agent reads the form data and either updates or creates a new Notes document.

When HTML is selected, the form data is sent back via a CGI variable called "Request_Content." The following data is URL encoded and has to be decoded to be viewed as plain text.

```
Name=John+Duggan;Car=Corvette%2e;Date=.....
```

A function included within the agent HTMLDecode is called that decodes the data.

```
Fields = doc.Request_Content(0)
Fields = HTMLDecode(fields)
```

Within the original Notes documents, there's a "computed when composed" field called DocID that stores the unique document ID. If the FDF form is submitted from the Web and the document does not contain this field, it means the FDF form is a new Notes document, otherwise it is an existing Notes document.

```
If Instr(fields,"DocID") <> 0 Then
  pos = Instr(1,fields,"DocID=")
  DocID = Mid$(fields,pos+Len("DocID="),32)
  Set newdoc = db.getdocumentbyUNID(DocID)
Else
    Set newdoc = db.createdocument
End If
```

Extract each individual field, its value, and place it into the Notes document.

```
pos =0
token="&"
pos2=Instr(pos+1,fields,token)

While (pos2<>0)
  token1=Mid(fields,pos+1,pos2-(pos+1))
  pos = pos2

  fieldname=Strleft(token1,"=")
  fieldvalue=Strright(token1,"=")
  Call newdoc.ReplaceItemValue( fieldname, fieldvalue)
  pos2=Instr(pos+1,fields,token)
Wend
```

Save the document and you now have a Notes document that has either been created or updated from an interactive PDF document!

Templates

If your users are using the full version of Acrobat, they can also use dynamic PDF forms. Acrobat allows a single page to be defined as a template, which can be used to dynamically generate a new form, or duplicate PDF pages dynamically.

More to it

FDF forms is a powerful technology, convenient for office environments still using paper forms. The existing paper forms can be scanned and converted to PDF. The electronic forms will be identical to their paper counterparts, thus retaining the organization's identity and branding while providing a quick process of putting paper documents online.

This article explains only a small bit about FDF forms with Domino, which together are capable of much more. Take the time to explore the possibilities further—it is well worth the effort. ■