More Flexible Reporting With XFRX

Doug Hennig

XFRX can make your reporting solutions more flexible since it allows you to output FRX reports to PDF, Microsoft Word, Microsoft Excel, and HTML files. This month, Doug takes a look at how easy it is to incorporate XFRX in your applications.

Visual FoxPro developers are becoming more and more interested in outputting reports to other destinations than just a printer or preview window. In the past couple of years, there's been a veritable explosion of tools that provide additional output choices for VFP reports. Some of these include FRX2Word by John Koziol, FRX2Any by Neova, and Mind's Eye Report Engine by Mind's Eye Inc. (which I reviewed in the May 2003 issue of FoxTalk). XFRX from Eqeus is another choice, and it's the subject of this month's column.

XFRX allows you to output FRX-based reports to PDF (without requiring a PDF writer such as Adobe Acrobat or Amyuni), Microsoft Word, Microsoft Excel, HTML, and other formats. It also provides a report preview container control you can use to preview reports in a VFP form.

Installing XFRX

You can download the evaluation version of XFRX from http://www.eqeus.com/xfrxdownload.php. The evaluation version is fully functional; it just adds a line of text at the bottom of each page that indicates it's a demo version.

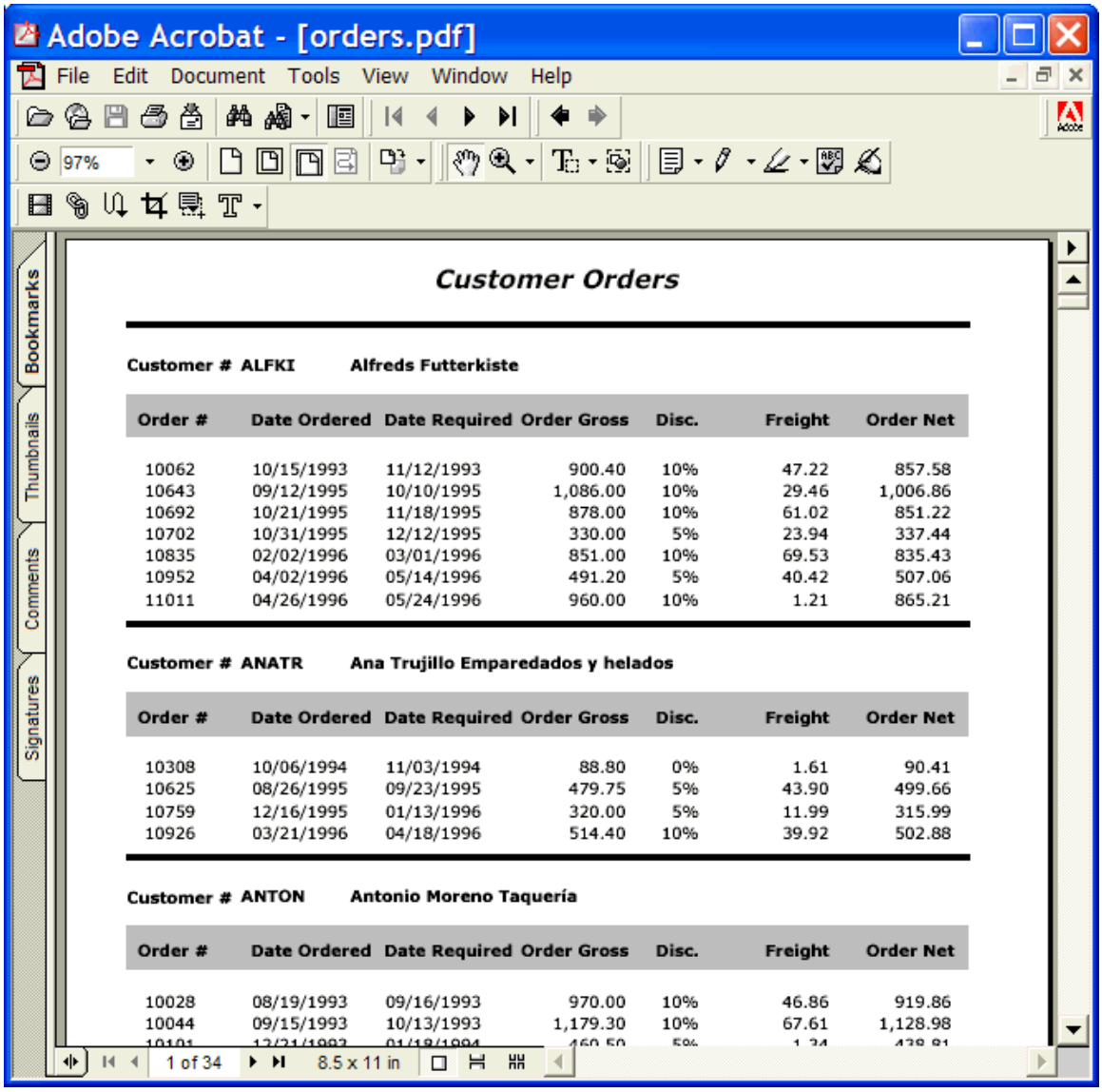
XFRX consists of three files: XFRX.APP, which contains the majority of the XFRX code; HNDLIB.DLL, which provides functions used internally by XFRX; and ZLIB.DLL, a freeware compression library. In addition, there are a number of files in the XFRXLIB subdirectory that comprise the preview container control. Several demo files are provided to show off the capabilities of XFRX and how to use them.

Using XFRX

Outputting an FRX with XFRX takes just a few lines of code. First, you call XFRX.APP with "XFRX#INIT" to return an XFRXSession object. You then call the SetParams method of this object to tell XFRX the name and type of the file to create, as well as some other settings. The ProcessReport method processes the report you specify as a parameter, and the Finalize method finishes the document creation process. Here's an example (taken from ORDERSPDF.PRG in the Subscriber Downloads for this month) that creates ORDERS.PDF from ORDERS.FRX. Figure 1 shows the resulting PDF file.

```
loSession = XFRX('XFRX#INIT')
lnStatus  = loSession.SetParams('orders.pdf', '', .F., ;
  '', .T., .F., 'PDF')
if lnStatus = 0
  loSession.ProcessReport('orders.frx')
  loSession.Finalize()
else
* handle error
endif lnStatus = 0
```

Figure 1. XFRX outputs PDF files from an FRX with only a few lines of code.



The SetParams method accepts the following parameters:

- The name of the document to create.
- The name of a directory in which to create temporary files (if not passed, the current directory is used).
- .F. to open the document after creation or .T. to suppress this behavior.
- The code page for the document (if not passed, CPCURRENT() is used).
- .F. to display processing messages or .T. to run in “silent” mode.
- .T. to open the document in a new session of Microsoft Word or .F. to use the current Word session (only used for Word output).
- The type of document to create: “DOC” for Word (this requires Word 2000 or later), “XLS” for Excel, “PDF” for PDF, “HTML” for HTML, “MHT” for MHT documents (I’m not sure what this is – it looks like HTML – and the documentation doesn’t really explain it), “CNT” to output a

report to a preview container object (discussed later), or "XFF" for a binary format you can load into the preview container.

The ProcessReport method accepts the following parameters (all but the first are optional):

- The name of the FRX file.
- The FOR clause for the report.
- .T. to use the SUMMARY clause.
- The report scope (such as "REST" or "NEXT 20").
- The WHILE clause.
- .T. to use "plain" format for HTML (no page breaks)

Since the parameters passed to the SetParams method specify the name and type of file to create, that's the only statement you need to change to output to a different file format. For example, changing two parameters in this method call in ORDERSPDF.PRG outputs to HTML rather than PDF:

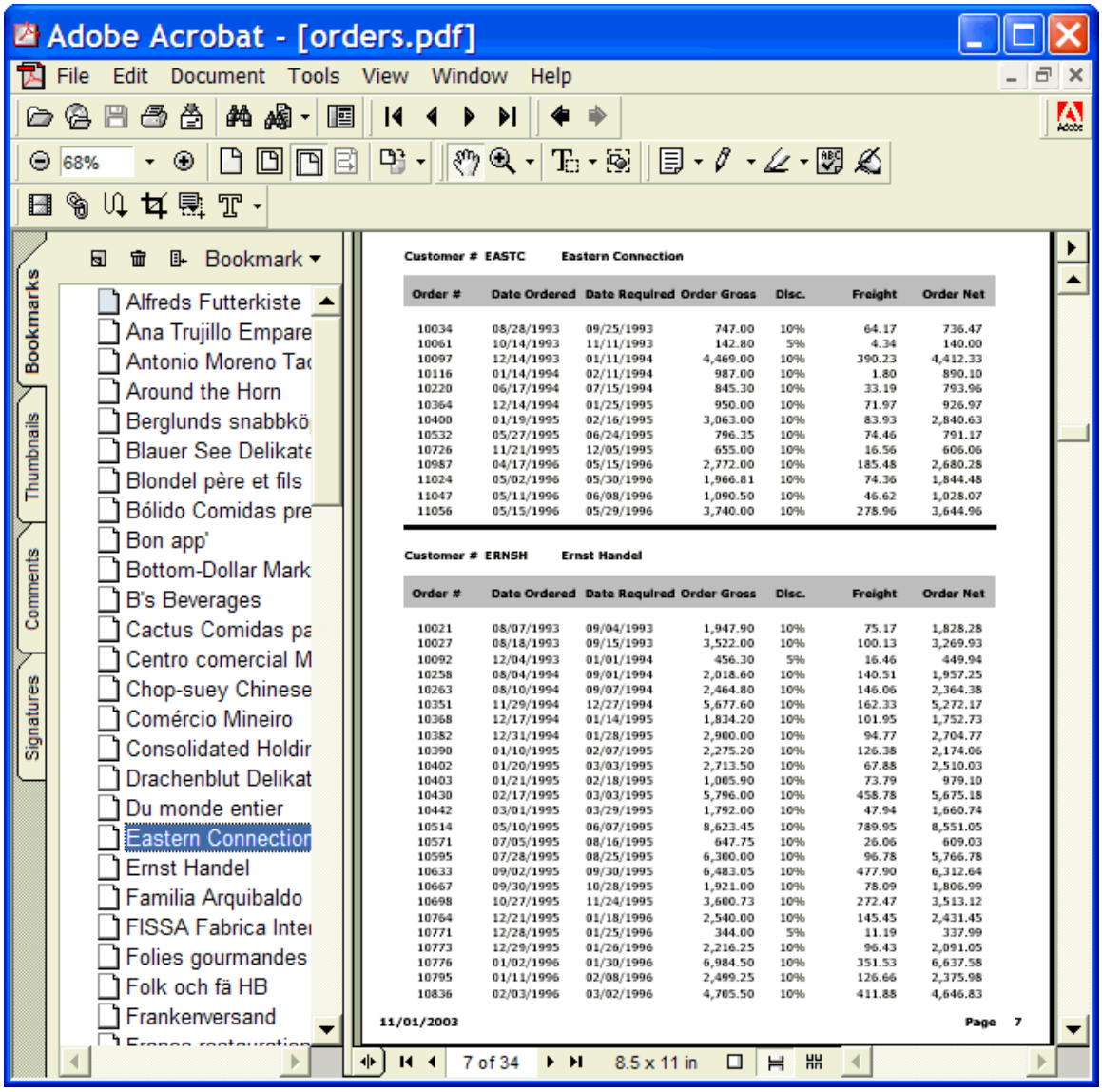
```
lnStatus = loSession.SetParams('orders.html', '', .F., ;
  '', .T., .F., 'HTML')
```

Getting fancier

One of the cool features in XFRX is the ability to do more than just output a report to file; using directives in the Comment of report objects, you can tell XFRX to create bookmarks and hyperlinks in the document.

Using bookmarks tells XFRX to generate a table of contents for the document. Figure 2 shows the orders report with bookmarks added for the company name. Clicking on an item in the table of contents displays that item in the main view. This feature is supported for both PDF and HTML output. In the case of HTML, XFRX actually generates three files: the one you specify, which defines two frames (one for the table of contents and the other for the actual content), one containing the table of contents, and one containing the report content.

Figure 2. XFRX automatically creates a table of contents when you use bookmarks.



It's very easy to create bookmarks: simply add #UR OUTLINE=*expression* into the Comment property of a report object, where *expression* is the expression used to create values for the table of contents. For example, ORDERBOOKMARKS.FRX is a clone of ORDERS.FRX except it has #UR OUTLINE=CUSTOMER.COMPANY for the Comment of the COMPANY field. The only other thing to do is to use loSession.SetOtherParams('PRINT_BOOKMARKS', .T.) if you're outputting to HTML (not needed with PDF). Here's an example, taken from ORDERSBOOKMARKS.PRG:

```
loSession = XFRX('XFRX#INIT')
lnStatus  = loSession.SetParams('orders.html', '', .F., ;
  '', .T., .F., 'HTML')
if lnStatus = 0
  loSession.SetOtherParams('PRINT_BOOKMARKS', .T.)
  loSession.ProcessReport('ordersbookmarks.frx')
  loSession.Finalize()
else
* handle error
endif lnStatus = 0
```

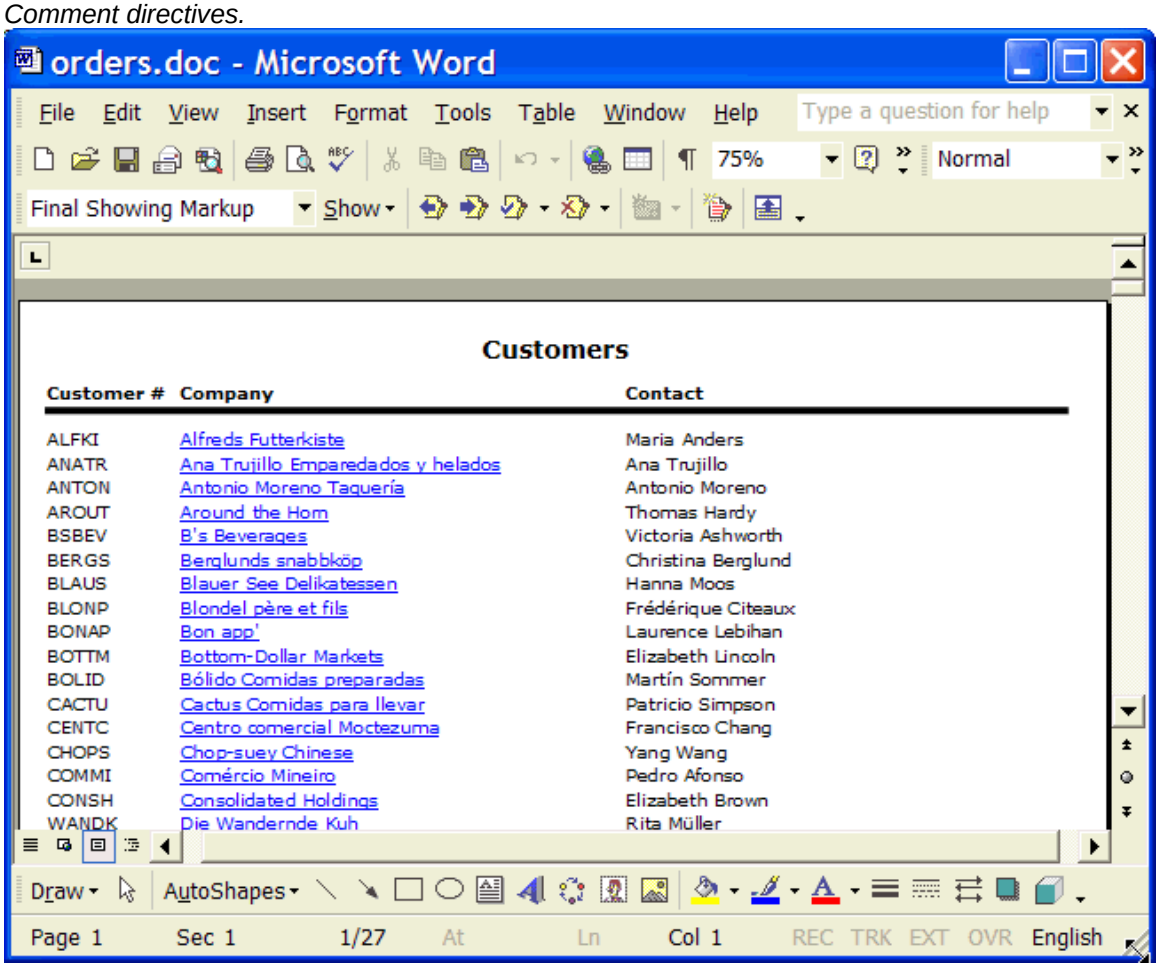
Creating hyperlinks is similar: you put directives into the Comment property of source and target objects to link them together. For source objects, use #UR A HREF=*destination*, where *destination* can be a

URL (e.g. http://www.stonefield.com) or a location in the output file (such as "#" + customer.company, which inserts the value of company name field). For target objects, use #UR A NAME=*destination*, where *destination* is the expression to use for the target name.

Here's an example that shows the use of this feature and another one: the ability to output multiple reports to a single file. CUSTOMERS.FRX is a simple report showing CUST_ID, COMPANY, and CONTACT from the CUSTOMERS table. The report object for the COMPANY field has #UR A HREF='#' + CUSTOMER.COMPANY for the Comment property so it hyperlinks to a target with the company name as the destination name. ORDERS.FRX also has an object for the COMPANY field, with #UR A NAME=CUSTOMER.COMPANY for its Comment. ORDERSHYPERLINKS.PRG outputs both of these reports to the same Word document (hyperlinking works with Word, HTML, and PDF), with CUSTOMERS.FRX acting as a different type of table of contents for ORDERS.FRX. Figure 3 shows the results.

```
loSession = XFRX('XFRX#INIT')
lnStatus  = loSession.SetParams('orders.doc', '', .F., ;
  '', .T., .F., 'DOC')
if lnStatus = 0
  loSession.ProcessReport('customers.frx')
  loSession.ProcessReport('orders.frx')
  loSession.Finalize()
else
* handle error
endif lnStatus = 0
```

Figure 3. You can hyperlink different sections of a report, or even different reports, using Comment directives.



If you output to an Excel document, you can specify the name of the worksheets using the SetOtherParams method, as this code taken from ORDERSEXCEL.PRG shows:

```
loSession = XFRX('XFRX#INIT')
lnStatus  = loSession.SetParams('orders.xls', '', .F., ;
  '', .T., .F., 'XLS')
if lnStatus = 0
  loSession.SetOtherParams('NEXT_SHEET_NAME', 'Customers')
  loSession.ProcessReport('customers.frx')
  loSession.SetOtherParams('NEXT_SHEET_NAME', 'Orders')
  loSession.ProcessReport('orders.frx')
  loSession.Finalize()
else
* handle error
endif lnStatus = 0
```

Previewing reports in a form

In addition to outputting to a file, XFRX also provides the ability to preview a report to a form using its own preview container control. Since this is all VFP code, you have more control over the appearance and behavior of the form than if you use the VFP preview window. To try this out, I created CUSTOMERS.SCX, dropped an XFCONT object (from XFRXLIB.VCX) on it, named the object oReport, and added the following code to the form's Init method:

```
local loSession, ;
  lnStatus
with This
  wait window 'Running report...' nowait
  .oReport.Reset()
  loSession = XFRX('XFRX#INIT')
  lnStatus  = loSession.SetParams(, , , , .T., , 'CNT')
  if lnStatus = 0
    loSession.SetOtherParams(.oReport)
    loSession.ProcessReport('customers.frx')
    loSession.Finalize()
  else
    return .F.
  endif lnStatus = 0
  wait clear
endwith
```

The "CNT" report type in the SetParams call tells XFRX that output should be prepared for the preview container control, and the SetOtherParams call attaches the preview container control to the session object.

Figure 4 shows what the preview form looks like when you run it. The preview container has several controls: scroll bars for the report page, page navigation controls (first, last, next, and previous pages), a scale factor combobox (sizes from 300% down to 10% as well as Fit Width and Fit in Window options), a button which shows or hides a pane for bookmarks, and a Find button which allows you to locate and highlight any text that appears in the report. You can also move around the report using the arrow keys, PgUp and PgDn keys, and from a shortcut menu.

Figure 4. The preview container control allows you to preview reports in a VFP form.

Other features

You can set properties of the created documents using the methods of the XFRXSession object shown in Table 1. Each method accepts the value for the property as a parameter. For example, to set the title of a document to “Customer Report”, use loSession.SetTitle(“Customer Report”).

Table 1. Use these methods to set properties of the documents XFRX creates.

Method	Applicable Document Type
SetAuthor	All
SetTitle	All
SetSubject	All
SetKeywords	All
SetCreator	PDF
SetProducer	PDF
SetComments	Word
SetCategory	Word
SetManager	Word
SetCompany	Word

Other methods set specific properties of certain types of files. The SetPasswords method allow you to assign passwords to Word and PDF documents. SetPermissions allows you to indicate what users can do to password-protected PDF files; you can control whether they can print the document, modify it, copy text or graphics, or add or modify annotations. The SetEmbeddingType method tells XFRX to embed fonts in the PDF file (either whole or font subsets).

If you set the fifth parameter of the SetParams call to .T., no message is displayed while XFRX processes the FRX file; otherwise, XFRX displays a simple “Processing” message during processing. If you want use another mechanism to show the progress of the processing, such as a thermometer bar, use the SetProgressObj method to attach an object that exposes an UpdateProgress method to the XFRXSession

object. Here's an example, ORDERSPROGRESSBAR.PRG, that uses the _Thermometer class provided with VFP as a progress meter. Since _Thermometer doesn't have an UpdateProgress method, I subclassed it to add that method (it calls the Update method to do the work). The "2" in the call to SetProgressObj tells XFRX to pass the percent complete to the UpdateProgress method; pass "1" to skip that parameter and only pass the page and report numbers.

```
loProgress = createobject('ProgressDialog')
loSession  = XFRX('XFRX#INIT')
lnStatus   = loSession.SetParams('orders.pdf', '', .F., ;
  '', .T., .F., 'PDF')
if lnStatus = 0
  loSession.SetProgressObj(loProgress, 2)
  loSession.ProcessReport('orders.frx')
  loSession.Finalize()
else
* handle error
endif lnStatus = 0

define class ProgressDialog as _thermometer of ;
  (home() + 'ffc\_therm.vcx')
  procedure UpdateProgress(tnReport, tnPage, tnPercent)
    if not This.Visible
      This.Show()
    endif not This.Visible
    This.Update(tnPercent, 'Processing page ' + ;
      transform(tnPage) + ' of report ' + ;
      transform(tnReport))
  endproc
enddefine
```

If you need more flexible PDF generation (such as generating PDF files from code rather than from FRX files), Eqeus also has a tool called PDF Library, which I didn't look at.

Pricing and licensing

XFRX is available in a number of configurations. Versions that output to only Word or HTML or only provide the preview container control are $99 each, versions that output to only PDF or Excel are $119 each, and a version that includes all of these is $269 (all prices are in US dollars). For source code, the prices are $199 for Word or HTML and $459 for the inclusive version. All versions are royalty-free, but you can't distribute source code (obviously). Free updates are provided for one year after purchase; after that, you can purchase updates as required. See http://www.eqeus.com for more details and current pricing.

Summary

XFRX makes short work of outputting FRX reports to PDF, Word, Excel, or HTML files. It only takes a few lines of code to impress your users with more flexible output options in your applications.

Doug Hennig is a partner with Stonefield Systems Group Inc. He is the author of the award-winning Stonefield Database Toolkit (SDT) and Stonefield Query, author of the CursorAdapter and DataEnvironment builders that come with VFP 8, and co-author of "What's New in Visual FoxPro 8.0", "What's New in Visual FoxPro 7.0", and "The Hacker's Guide to Visual FoxPro 7.0", from Hentzenwerke Publishing. Doug has spoken at every Microsoft FoxPro Developers Conference (DevCon) since 1997 and at user groups and developer conferences all over North America. He is a Microsoft Most Valuable Professional (MVP) and Certified Professional (MCP). Web: www.stonefield.com and www.stonefieldquery.com Email: dhennig@stonefield.com