Making the Most of VFP 9 SP2 Reports (Part I)

Cathy Pountney
Memorial Business Systems, Inc.
cathy@frontier2000.com
www.frontier2000.com
www.mbs-intl.com
www.cathypountney.blogspot.com
Twitter: frontier2000

In addition to lots of bug fixes, VFP 9 SP2 offered up some major enhancements to the Visual FoxPro Report Writer. In this session (the first of a 2-part series), we explore the new Dynamics, Advanced, and Rotation features. We also go behind the scenes and learn how these features are implemented. If you want to write professional reports with lots of pizzazz, this session is a must-see.

Installing SP2

One would think installing a Service Pack is a straight-forward and simple task. However, with VFP 9 SP2, that's not the case. Special care needs to be taken to prepare your system for the installation of SP2. After the installation, there are additional steps you need to follow to be sure you have all the bug fixes installed.

Preparing to Install SP2

Due to a bug in the VFP 9 SP2 installation script, it's possible that SP2 doesn't install correctly on your machine. Even worse, it doesn't notify you that it didn't install correctly, so you think you have SP2. Yet when you try to use the new Report Writer features, nothing works and you're left scratching your head and wondering what the heck you've done wrong.

Installing SP2 on top of SP1 is the trigger that makes the installation bug rear its ugly head. To get around this, simply uninstall VFP 9 completely and install SP2 on top of a fresh VFP 9 install. If you already have a fresh VFP 9 install without any service packs, you're good to go.

For Vista, Windows 7, and Server 2008 users, another problematic issue can occur during the installation of VFP 9 SP2 due to the Virtual Store. Files sitting in the "Program Files" directory are not read-write, except during installation. After that point, any attempt to write to a file in "Program Files" is *virtualized* by the O/S. Virtualization means the O/S makes a copy of the file in another folder called "\Users\[UserName]\AppData\Local\VirtualStore". Your changes are saved in this directory, not the original "Program Files" directory. Any future access to this file routes to the copy sitting in the Virtual Store, not the original file in the "Program Files" directory.

So why is virtualization a problem for the SP2 installation? Well, it has to do with the fact that FoxPro actually writes to a VCX the moment you open it and view it. It doesn't matter if you changed anything or not. Just the mere task of viewing the VCX causes a write to the file which means the O/S makes a copy of that VCX in the Virtual Store. During the SP2 installation, several VCX files in the FFC directory ("\Program Files\Microsoft Visual FoxPro 9\FFC") are overwritten with new versions. However, if you have previously opened any VCXs that reside in the FFC directory, they have been virtualized. Now the new SP2 version is in "Program Files", but the older version is sitting in the Virtual Store. Unfortunately, it's the older version in the Virtual Store that gets accessed from this point forward, not the new version sitting in Program Files.

The solution to the virtualization problem is to delete any VFP files from the Virtual Store before you install SP2. However, if you have purposefully made changes to any of those files, you will need to rename them, install SP2, and then redo your changes in the updated files. Also, remember there is a different Virtual Store for each Windows user so be sure to delete the Virtual Store for all users.

Installing SP2

Once you've addressed the SP1 and virtualization issues, you're ready to install SP2. If you had to remove VFP 9, go ahead and install it now. If you've already got it installed, track down your registration key, because you're going to need it. Next, download SP2 from http://msdn.microsoft.com/en-us/vfoxpro/default.aspx and simply run it.

There are some other SP2-related files that you can optionally download from the same URL mentioned above:

- Help download for Visual FoxPro 9.0 SP2
- XSource for Visual FoxPro 9.0 SP2
- Visual FoxPro 9.0 "Sedna" AddOns
- Microsoft OLE DB Provider for VFP 9.0 (SP2 update)

Beyond SP2

When VFP 9 SP2 was first released, a nasty Report Writer bug was quickly discovered with Data Groups. You can read more about this bug on my blog (http://cathypountney.blogspot.com/2007/11/gotcha-serious-report-bug-with-data.html). The good news is that a hotfix is available to fix this bug. Even better news is the hotfix is available on the MSDN Code Gallery so you don't have to go through the cumbersome process of calling Microsoft Product Support and opening a support call. The hotfix can be downloaded from here: http://code.msdn.microsoft.com/KB968409.

There's another minor bug that causes SET TALK to be on during a report run which spews lots of undesirable text on the screen. This bug is in one of the FFC_ReportListener.vcx classes and you can easily fix it yourself. The steps to fix it are described on my blog here: http://cathypountney.blogspot.com/2009/04/set-talk-appears-to-be-on-when-running.html. For Vista, Windows 7, and Server 2008 users, be sure to pay attention to the special instructions to avoid problems with virtualization.

Looking towards the future, VFPX is the place to go (http://www.codeplex.com/Wiki/View.aspx?ProjectName=VFPX). VFPX is a Community supported project on CodePlex. FoxPro developers volunteer their time to create tools, utilities, and other applications that help other FoxPro developers. In addition, the XSource and FFC (FoxPro Foundation Classes) code delivered with Visual FoxPro is now maintained on VFPX. This allows the FoxPro Community to fix bugs and add enhancements. For example, the help file has been greatly improved on VFPX and I highly recommend you download and use the VFPX version instead of the one provided by Microsoft.

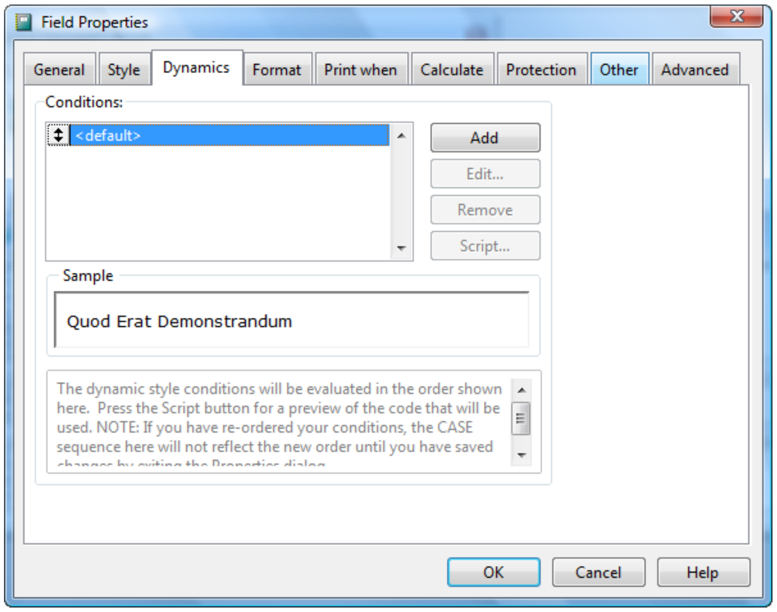
In Making the Most of VFP 9 SP2 Reports – Part II, I'll introduce a new VFPX project to share enhancements and special effects to use with reports. I also encourage everyone to get involved in some way with one or more projects on VFPX. Project Managers are always looking for developers to volunteer their time. If you don't want to code, volunteer to test or write documentation for an existing project. There's always something you can do!

Dynamics

The most visible improvement made to VFP 9 reporting is the addition of a new Dynamics tab on the Property dialog for Fields, Rectangles, and Picture objects. This tab lets you change the text and format of Field objects and the height and width of Rectangle and Picture objects dynamically. This opens the door for lots of flexibility and gets rid of that cumbersome practice of having to stack lots of objects with the same expression on top of each other, each one with different format options and different Print When logic.

It’s important to note that to take advantage of the new dynamics features, your reports must be run using a Report Listener derived from one of the classes in the FFC directory. I talk more about that later in the section “Running reports with dynamics,” but for now, let’s learn how to define the dynamics.

Figure 1 shows the Dynamics tab for a Field object. You can create multiple conditions that apply to any one report object. A list of defined conditions appears in the listbox on the left. The first entry in the listbox is always “<default>” which represents the text when none of the defined conditions are true. The remaining items in the listbox are the conditions that have been defined for this object and the order of the items in the list is extremely important. Behind the scenes, VFP creates a CASE/ENDCASE structure for the list of defined conditions. As soon as one condition evaluates to true, that condition is applied to the report object and no other conditions are evaluated or applied.



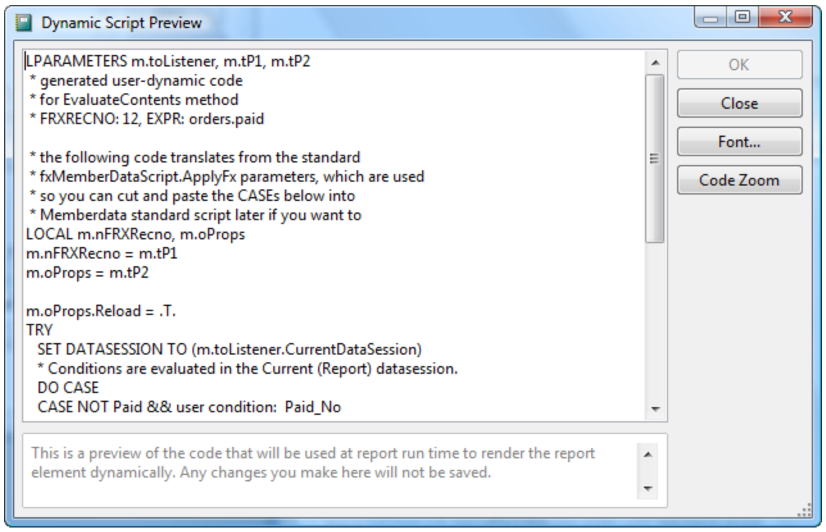
Quod Erat Demonstrandum

The dynamic style conditions will be evaluated in the order shown here. Press the Script button for a preview of the code that will be used. NOTE: If you have re-ordered your conditions, the CASE sequence here will not reflect the new order until you have saved

Figure 1. Use the Dynamics tab to provide flexibility in formatting objects.

The Add button prompts for a condition name and then invokes a dialog for entering the condition. The Edit button invokes the dialog for editing the highlighted condition. The Remove button asks you to confirm your intentions and then deletes the highlighted condition.

The Script button invokes a preview window (shown in **Figure 2**) to show the code generated behind the scenes for all the conditions of this report object. The Close button closes the preview and returns you to the Dynamics tab. The Font button invokes the Font dialog giving you the opportunity to change the font of the code shown in the Script Preview window. The Code Zoom button invokes a code editor window with the same code shown in the Script Preview window. The difference is that in the Script Preview window the code is entirely black whereas the code in the Code Zoom window is color-coded just as if you were editing the code in a VFP program or class.



```
LPARAMETERS m.toListener, m.tP1, m.tP2
 * generated user-dynamic code
 * for EvaluateContents method
 * FRXRECNO: 12, EXPR: orders.paid

 * the following code translates from the standard
 * fxMemberDataScript.ApplyFx parameters, which are used
 * so you can cut and paste the CASEs below into
 * Memberdata standard script later if you want to
LOCAL m.nFRXRecno, m.oProps
m.nFRXRecno = m.tP1
m.oProps = m.tP2

m.oProps.Reload = .T.
TRY
  SET DATASESSION TO (m.toListener.CurrentDataSession)
  * Conditions are evaluated in the Current (Report) datasession.
  DO CASE
  CASE NOT Paid && user condition:  Paid_No
```

This is a preview of the code that will be used at report run time to render the report element dynamically. Any changes you make here will not be saved.

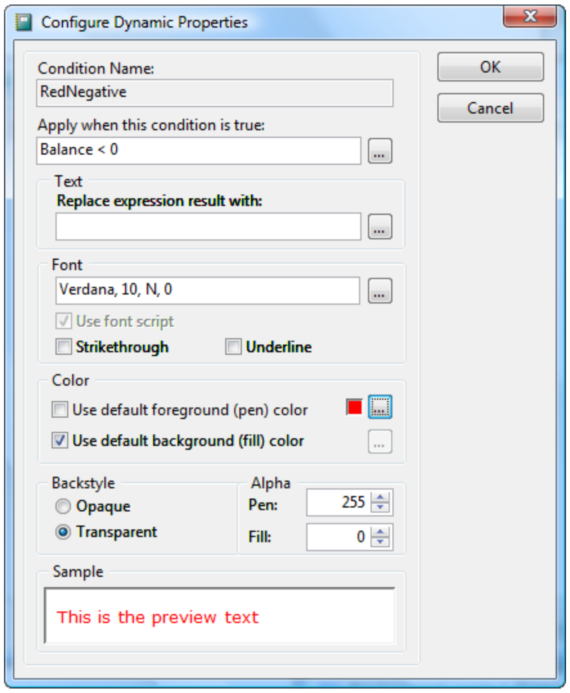
Figure 2. Use the Dynamic Script Preview window to show the code being used behind the scenes.

It's worth noting that both the Dynamic Script Preview window and the Code Zoom window appear as if they are read-write, however, this is just an illusion. If you change code in the Dynamic Script Preview, nothing is saved when you close the dialog. If you change code in the Code Zoom window, upon closing you'll get a prompt asking if you want to save the changes. However, this is also an illusion because the file name is just a temporary file so saving has no permanent effect.

The Sample region shows how the text appears on the report when the highlighted condition (or default) is true.

Field objects

How many times have you wanted negative numbers to appear in red? Prior to SP2, you had to add two Field objects to the report with one lying on top of the other. One was defined as black with a Print When condition for the positive values. Another Field object was defined as red with a Print When condition for negative values. In SP2 you can create a dynamic condition to solve this need as shown in **Figure 3**.



Condition Name: RedNegative

Apply when this condition is true: Balance < 0

Text

Replace expression result with:

Font

Verdana, 10, N, 0

☑ Use font script

☐ Strikethrough ☐ Underline

Color

☐ Use default foreground (pen) color 🟥

☑ Use default background (fill) color

Backstyle: Opaque [✓ Transparent]

Alpha: Pen: 255 Fill: 0

Sample

This is the preview text

Figure 3. Use the Configure Dynamics Properties dialog for a Field object to dynamically change formatting options.

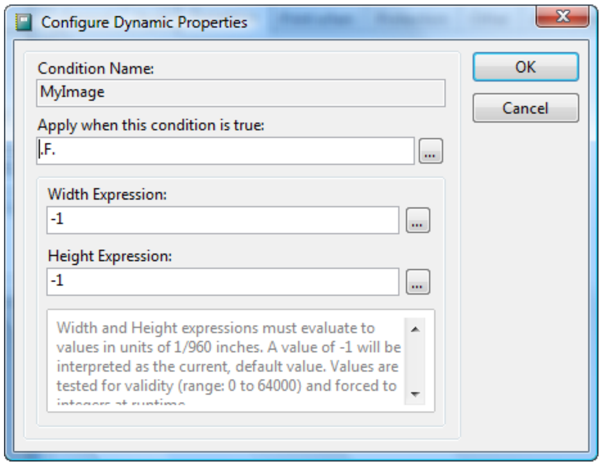
The first piece of information to enter is the condition. It's an expression; when it evaluates to true, this condition is applied to the Field object. The second item to enter is a replacement for the text being printed. In many cases, this item doesn't apply and you can leave it blank. If, however, you do want to change the actual text being printed, this gives you the option to override what appears on the report. For example, you can wrap the field value with parenthesis when the value is negative. All fields of the cursors and tables available to the report, along with all variables available to the report, can be used in both the condition and the replacement text.

The next sections of the dialog allow you to change the font properties, the color properties, and the backstyle properties. At the time the dynamic condition is created, the properties of the object are copied into the properties of the dynamic condition. This is important to remember because down the road, if you change the format properties of the original Field object, none of the defined conditions are affected. You should review each of the conditions and determine whether changes are required for each one.

The final area of the dialog shows an example of how the text appears when this condition is applied.

Rectangle and Image objects

With Rectangle and Image objects, the properties that can be dynamically changed are different than Field objects (see **Figure 4**.) Similar to Field objects, the first item to fill in is the condition to evaluate. The second and third items to fill in are the width and height expressions, respectively. A value of -1 tells Visual FoxPro to use the height or width already defined for the object. Any other value overrides the defined dimension when this condition evaluates to true. The expected unit of measure is 1/960 of an inch (regardless of whether the ruler setting is inches or metric.)



Condition Name: MyImage

Apply when this condition is true: .F.

Width Expression: -1

Height Expression: -1

Width and Height expressions must evaluate to values in units of 1/960 inches. A value of -1 will be interpreted as the current, default value. Values are tested for validity (range: 0 to 64000) and forced to

Figure 4. Use the Configure Dynamic Properties dialog for Rectangle and Image objects to dynamically change the width and height of objects.

One time you may want to use this feature is when you have image objects defined in the detail band on a report, but not all records will have images. If you've ever attempted to do this, you probably realized fairly quickly there's a big flaw. The easy part is using Print When logic combined with Remove line when blank to suppress the images when they're not needed. That part works just great.

The part that causes a problem is the internal Report Engine's logic for determining when a new page is needed. Prior to printing a band, VFP looks at the defined height of the band and if there isn't enough room, a new page is invoked. Visual FoxPro does not bother to evaluate the Print When logic of each of the objects in the band before it determines if there's enough room. That means it assumes the image is going to print and often this results in a lot of wasted space at the end of each page.

With the dynamic feature in SP2, the wasted space can be avoided. Start by defining the image with very small width and height values. Next, create a dynamic condition whose condition expression is the same as the Print When logic. Then change the width and height properties to be the desired size. When the report runs, if the condition evaluates to true, VFP expands the size of the image. It's also smart enough to handle moving to the next page in the situation where the changed height causes the bigger image to overrun the end of the page. On the flip side, because the defined height of the detail band is much shorter, wasted space doesn't happen when the image doesn't print.

This approach works pretty good but falls short if there are other objects in the band that print before the picture. What can happen is the other objects print near the bottom of the page. Then VFP determines the newly expanded picture doesn't fit so the picture gets moved to the next page. Now the objects in the band are split between two pages.

To overcome this final obstacle, add a rectangle that spans the entire width of the report at the very top of the band. Make the height the minimum height possible. Mark all the other objects in the band as float so they move down if the rectangle grows taller. Finally, add some dynamic logic to calculate how much room is needed for the entire band if the picture is expanded. Compare that value against m.oProps.MaxHeightAvailable to determine if there's enough room to print the entire band. If not, set the new Height to m.oProps.MaxHeightAvailable + 1 to force the shape, and all the other objects in the band, to the next page. This technique is demonstrated in the 020_dynamics_shapes_5.frx report included with this session. Note that you should change the pen color of the shape to white in your production application. It's green in the sample to make it easy to see the concept.

Running reports with dynamics

If you run a report using the FoxPro base ReportListener class, you won't see any of the dynamics you defined in the report. Similarly, if you created your own Report Listener class derived from the FoxPro base ReportListener class, you won't see any of the dynamics with that either. And if you use the _ReportListener class in the _ReportListener class library of the FFC directory, or you use one of your Report Listener classes derived from _ReportListener, you still won't get the dynamics.

```
*-- This doesn't work
loRL = CREATEOBJECT('ReportListener')
loRL.ListenerType = 1 && Preview
REPORT FORM 010_dynamics_fields OBJECT m.loRL

*-- This doesn't work either
loRL = NEWOBJECT('_ReportListener', HOME() + '\FFC\_ReportListener')
loRL.ListenerType = 1 && Preview
REPORT FORM 010_dynamics_fields OBJECT m.loRL
```

So what in the world do you have to do? The trick is to understand how the new dynamics feature is implemented. It is not implemented in the core Report Engine. Instead, it's implemented through a new Report Listener class called fxListener defined in the _ReportListener class library which resides in the FFC directory. That means you have to run the report using a Report Listener derived from fxListener or use fxListener itself.

```
*-- This works
loRL = NEWOBJECT('fxListener', HOME() + '\FFC\_ReportListener')
loRL.ListenerType = 1 && Preview
REPORT FORM 010_dynamics_fields OBJECT m.loRL
```

See, it's not that hard once you know the trick! Actually, Microsoft put together a mechanism behind the scenes that instantiates the right Report Listener class for you so you don't have to worry about it. A system variable called _ReportOutput contains the name of an application that determines which Report Listener to run. This system variable defaults to the application Microsoft ships with VFP 9. Using a traditional REPORT FORM command like the following, invokes the _ReportOutput application which instantiates the fxListener class.

```
*-- Traditional syntax
SET REPORTBEHAVIOR TO 90
REPORT FORM 010_dynamics_fields PREVIEW
```

You can also use the TYPE clause to tell the _ReportOutput application which Report Listener to instantiate, as shown in the following code.

```
*-- Using the TYPE clause
REPORT FORM 010_dynamics_fields OBJECT TYPE 1 && Preview
REPORT FORM 010_dynamics_fields OBJECT TYPE 4 && XML
REPORT FORM 010_dynamics_fields OBJECT TYPE 5 && HTML
```

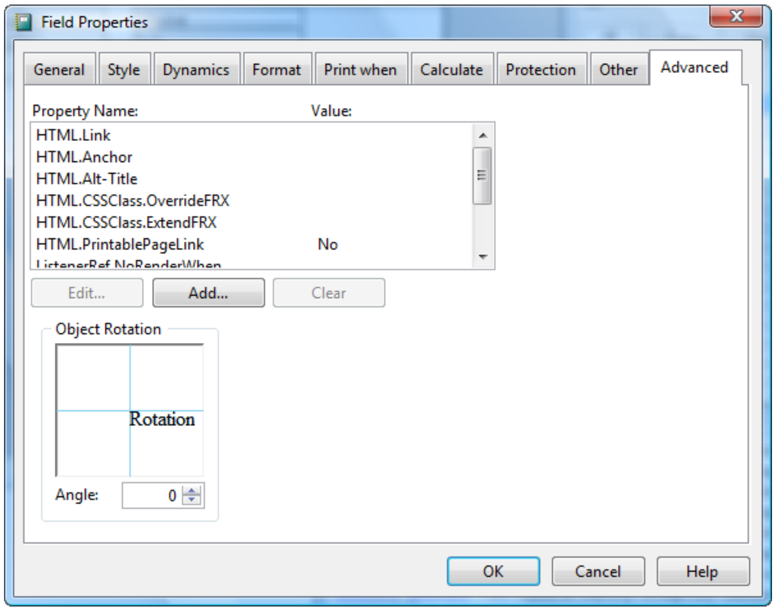
Just like many other things in Visual FoxPro, there's more than one way to do the same thing. The following shows another example of how to run a report and let the _ReportOutput application instantiate the appropriate Report Listener.

```
*-- Instantiate a Listener
LOCAL loRL
DO (_ReportOutput) WITH 1, loRL && 1 = Preview, 4 = XML, 5 = HTML
REPORT FORM 010_dynamics_fields OBJECT m.loRL
```

The important thing to remember is that if you want to take advantage of these new features, the right Report Listener class must be used. You can create your own Report Listener classes if you want, as long as you remember to subclass them from fxListener or something farther down the hierarchy. The other option is to let the _ReportOutput application instantiate the appropriate Report Listener provided with SP2.

Advanced Attribute Properties

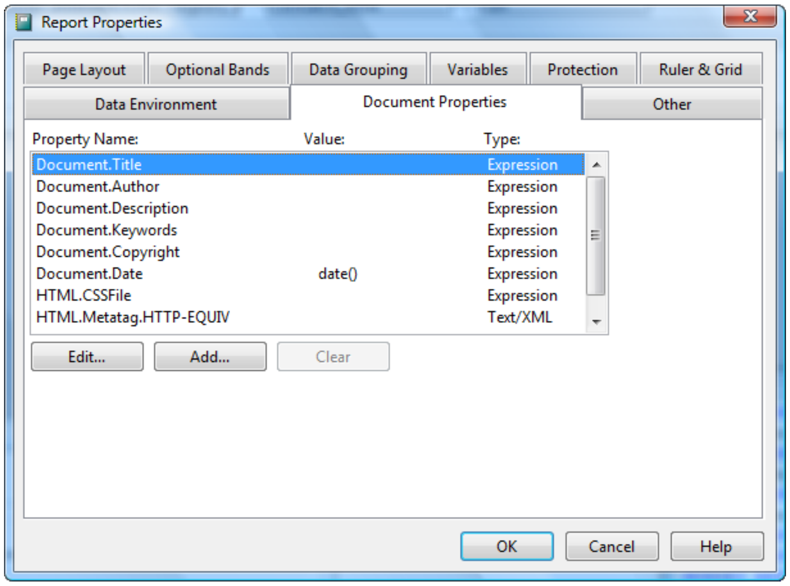
Another new feature added in SP2 is the new Advanced tab available on the various report object Properties dialogs (see **Figure 5**) and the new Document Properties tab available on the Report Properties dialog (see **Figure 6**). These new tabs provide access to the new advanced attribute properties feature which gives you even more ways to manipulate reports and the objects on the reports.



Property Name:	Value:
HTML.Link	
HTML.Anchor	
HTML.Alt-Title	
HTML.CSSClass.OverrideFRX	
HTML.CSSClass.ExtendFRX	
HTML.PrintablePageLink	No
ListenerRef.NoRenderWhen	

Angle: 0

Figure 5. Use the Advanced tab to manipulate objects on reports.



Property Name:	Value:	Type:
Document.Title		Expression
Document.Author		Expression
Document.Description		Expression
Document.Keywords		Expression
Document.Copyright		Expression
Document.Date	date()	Expression
HTML.CSSFile		Expression
HTML.Metatag.HTTP-EQUIV		Text/XML

Figure 6. Use the Document Properties tab to manipulate a report.

Both tabs work in a similar manner. The grid shows a list of attribute properties that are configurable for the report or the selected report object. The first column shows the Property Name and the second column shows the value currently assigned to the attribute property. The grid on the Document Properties tab has a third column which shows the type of value that can be assigned to the attribute property.

Highlight an attribute property and select the Edit button to change the value. What happens next depends on the type of value the selected attribute property expects.

- **Expression:** The standard Visual FoxPro Expression Builder dialog appears so you can enter a valid expression.
- **Text/XML:** A new dialog appears giving you the opportunity to enter any text or XML string.
- **String:** A dialog, the Visual FoxPro InputBox(), appears giving you the opportunity to enter a simple string.
- **File:** A standard Windows File Open dialog appears so you can select a valid file.
- **Yes/No:** The current value is toggled between Yes and No.

The Add button gives you the opportunity to define your own advanced attribute properties which are discussed in my Part II counterpart session. The Clear button removes any value you assigned to an attribute property and restores the default value, if one is defined.

It's important to note that, similar to the dynamics feature, the advanced attribute properties feature is also implemented using the fxListener class.

The advanced attribute properties included in Service Pack 2 can be broken up into four different categories: rendering, rotation, HTML, and document.

Rendering-related attribute properties

Two of the attribute properties on the Advanced tab give you the ability to suppress the rendering of an object.

- **ListenerRef.NoRenderWhen** (Expression)**:** If the value of this attribute property evaluates to true, this object is not rendered in the output. The expression is evaluated each time the object is about to be rendered. Use this when the rendering condition is dynamic and changes based on each record in the report.
- **ListenerRef.PreProcess.NoRenderWhen** (Expression)**:** If the value of this attribute property evaluates to true, this object is not rendered in the output. The expression is evaluated only once during the LoadReport method of the Report Listener. Use this when the rendering condition does not change throughout the life of the report. For example, you might use this attribute property to control the printing of the phrase "Draft" on a preliminary financial report, yet suppress the phrase during the printing of the final version of the financial report.

The ListenerRef.NoRenderWhen attribute property differs slightly from Print When logic in when it is evaluated. As you may already know, there is a bug in Visual FoxPro 9.0 that can cause the Print When logic to be evaluated against the wrong record under certain conditions (http://fox.wikis.com/wc.dll?Wiki~VFP9SP2BugList~Wiki). You can use the ListenerRef.NoRenderWhen attribute property to work around this bug. Simply move your Print When logic into this attribute property and run the report with the fxListener class. However, there are two things worth mentioning about this workaround. First, you must reverse the logic when moving an expression from Print When into the ListenerRef.NoRenderWhen attribute property because Print When describes when you want to print the object and NoRenderWhen describes when you do NOT want to print the object. The other issue is that using the ListenerRef.NoRenderWhen attribute property does not offer the "Remove if blank" option the Print When option does.

The ListenerRef.PreProcess.NoRenderWhen attribute property differs from Print When because it is only evaluated once at the beginning of the report run instead of each time the object is processed. When this attribute property is used, the Report Listener creates a temporary copy of the FRX. It then evaluates this attribute for each object and removes any objects for which this attribute evaluates to .T. from the temporary FRX cursor. Obviously, this improves the performance of the report because it doesn't have to handle the object for every record in the driving cursor.

Use of these rendering attribute properties requires the use of the fxListener class or a class derived from it. At the beginning of the report run, the fxListener class queries the FRX to see if any of the rendering attribute properties are used. If it finds any, it calls the AddCollectionMember method to instantiate the gfxNoRender class and registers it with the GFXs collection. The gfxNoRender class resides in the _ReportListener class library of the FFC and contains all the code that handles implementing these attribute properties.

Rotation attribute property

Rotating objects on a report is something we developers have wanted for a long time. Microsoft finally answered our requests in Visual FoxPro 9.0 Service Pack 2. All report objects (labels, fields, lines, rectangles, and pictures) can be rotated. Use the Advanced tab to enter the rotation angle. Enter a value of 0 to 359 or click in the Object Rotation box to indicate the rotation. Keep in mind the rotation direction is clockwise and the pivot point is the upper-left corner of the object. **Figure 7** shows examples of how various objects rotate. The object in the bottom-right corner of each cross-hair is the original object without any rotation. The other object has the rotation angle indicated below the object. The pivot point is the middle of the cross-hair.

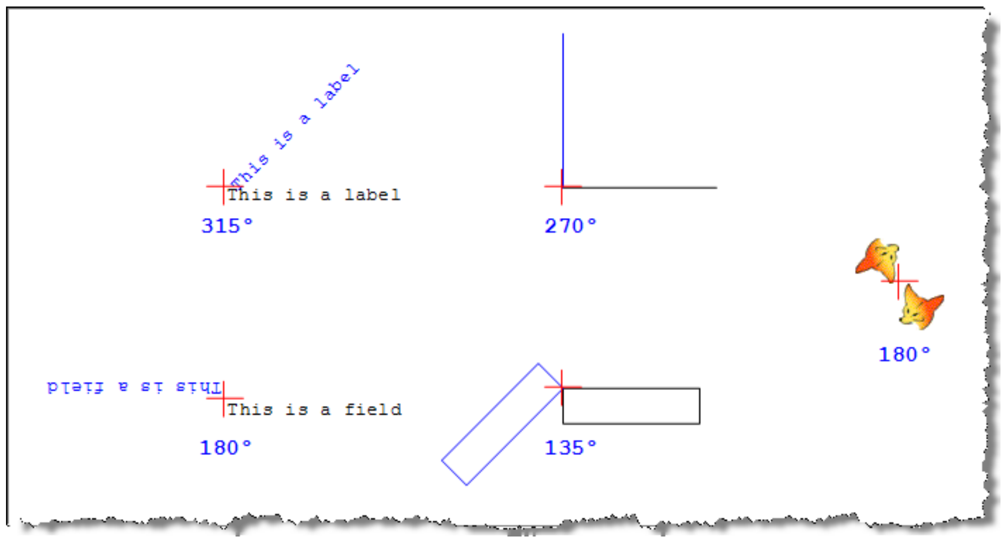


Figure 7. Rotation is a long-awaited feature finally provided by Visual FoxPro 9.0 Service Pack 2.

Use the fxListener class or a class derived from it to take advantage of rotation. At the beginning of the report run, the fxListener class queries the FRX to see if any objects have rotation indicated. If it finds any, it calls the AddCollectionMember method to instantiate the gfxRotate class and registers it with the GFXs collection. The gfxRotate class resides in the _ReportListener class library of the FFC and contains all the code that handles rotating objects.

HTML-related attribute properties

Several attribute properties on the Document Properties tab of the report and the Advanced tab of report objects relate to HTML output. Use of these attribute properties requires the report to be run using the HTMLListener class in the _ReportListener class library in the FFC directory, or a class derived from the HTMLListener class. These advanced attribute properties correspond with various pieces of an XSLT Transform for generating HTML output.

Document attribute properties (in the report's Document Properties tab)

- **HTML.CSSFile** (Expression)**:** Determines the Cascading Style Sheet used by the HTML output.
- **HTML.Metatag.HTTP-EQUIV** (Text/XML)**:** Determines the metatag information used to generate additional header information in the HTML output.
- **HTML.TextAreasOff** (Yes or No)**:** Determines whether the <TextArea> tag is used in the HTML output.

Object attribute properties (in an object's Advanced tab)

- **HTML.Link** (Expression)**:** Indicates the hyperlink inserted into the HTML output for this object. This attribute property is not available for Shape or Image objects.

- **HTML.Anchor** (Expression)**:** Indicates the anchor tag inserted into the HTML output.
- **HTML.Alt-Title** (Expression)**:** Indicates the alternate title inserted into the HTML output.
- **HTML.CSSClass.OverrideFRX** (Expression)**:** Indicates a Cascading Style Sheet class that is used for this object instead of what is defined as the default XSLT in the HTMLListener class.
- **HTML.CSSClass.ExtendFRX** (Expression)**:** Indicates a Cascading Style Sheet class that is used in addition to what is defined as the default XSLT in the HTMLListener class.
- **HTML.PrintablePageLink** (Yes or No)**:** Indicates whether this object has a hyperlink to a printable page. This attribute property is not available for Shape or Image objects.

Using the HTML-related attribute properties

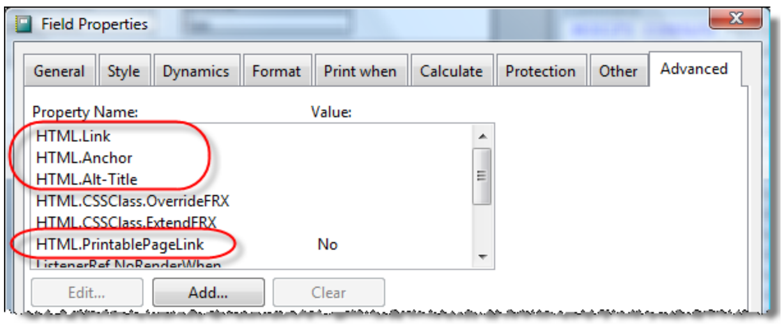
The following code shows two examples of how to run a report with the HTMLListener class. The first example directly instantiates the HTMLListener class and the second example uses indirect instantiation.

```
*-- Direct Example
LOCAL loRL
loRL = NEWOBJECT('HTMLListener', HOME() + 'FFC\_ReportListener')
loRL.TargetFileName = 'c:\MyTests\TestHTML'
REPORT FORM 040_adv_html OBJECT loRL

*-- Indirect Example
LOCAL loRL
DO (_ReportOutput) WITH 5, loRL
loRL.TargetFileName = 'c:\MyTests\TestHTML'
REPORT FORM 040_adv_html OBJECT loRL
```

Link, Anchor, Alt-Title and PrintablePageLink example

Figure 8 shows the HTML.Link, HTML.Anchor, HTML.Alt-Title and HTML.PrintablePageLink attribute properties on the Advanced tab for an object.



Property Name:	Value:
HTML.Link	
HTML.Anchor	
HTML.Alt-Title	
HTML.CSSClass.OverrideFRX	
HTML.CSSClass.ExtendFRX	
HTML.PrintablePageLink	No
ListenerRef.NoRenderWhen	

Figure 8. Use the Advanced tab to set the HTML.Link, HTML.Anchor, HTML.Alt-Title and HTML.PrintablePageLink attribute properties.

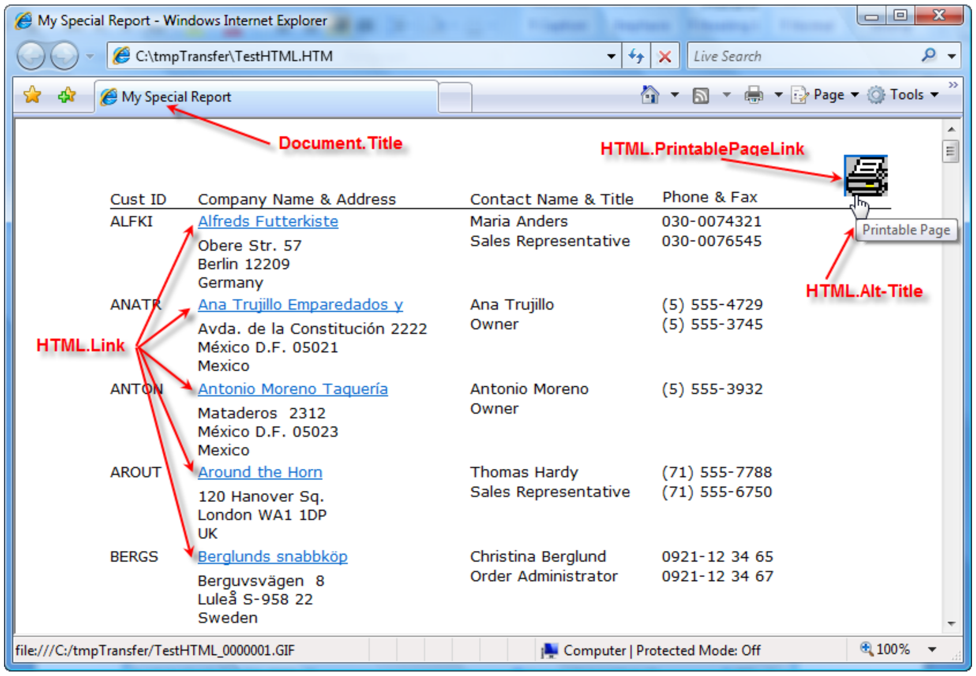
Use the HTML.Link attribute property to assign a hyperlink to any particular object on a report. You might specify the name of a field in the table that holds a URL address. Be sure to include a ":" in the string to avoid having a path added to the beginning of the string. For example, enter an expression such as: "http://www." + SomeTable.SomeField.

Use the HTML.Anchor attribute property to assign an anchor tag to any particular object on a report. An anchor tag is a bookmark that you can easily jump to by using "#" plus the name of the anchor tag. For example, on a customer report you can set the HTML.Anchor attribute property of the customer object to the customer_id field. This results in a bookmark being set for each customer on the report.

Use the HTML.Alt-Title attribute property to assign the text that displays when you hover over an object on the HTML output. One example is to show the URL address when you hover over a hyperlink. Another example is to show additional information that doesn't fit on the report. Keep in mind that for visually impaired users, this is the text that is read out loud by screen reader programs.

Use the HTML.PrintablePageLink attribute property to instruct the Report Listener to generate a GIF file for each page of the report. The object with the HTML.PrintablePageLink attribute property is clickable on the HTML output. Clicking the object displays the GIF file related to the page the object is on.

Figure 9 shows an example of HTML output using HTML.Link, HTML.Alt-Title and HTML.PrintablePageLink attribute properties. It also demonstrates the Document.Title property discussed later in the section "Document-related attribute properties."

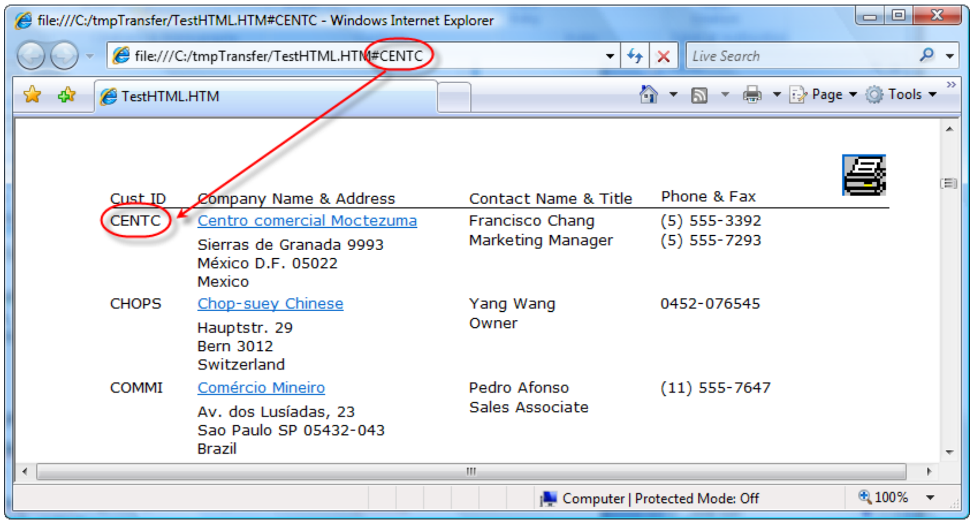


Cust ID	Company Name & Address	Contact Name & Title	Phone & Fax
ALFKI	Alfreds Futterkiste Obere Str. 57 Berlin 12209 Germany	Maria Anders Sales Representative	030-0074321 030-0076545
ANATR	Ana Trujillo Emparedados y Avda. de la Constitución 2222 México D.F. 05021 Mexico	Ana Trujillo Owner	(5) 555-4729 (5) 555-3745
ANTON	Antonio Moreno Taquería Mataderos 2312 México D.F. 05023 Mexico	Antonio Moreno Owner	(5) 555-3932
AROUT	Around the Horn 120 Hanover Sq. London WA1 1DP UK	Thomas Hardy Sales Representative	(71) 555-7788 (71) 555-6750
BERGS	Berglunds snabbköp Berguvsvägen 8 Luleå S-958 22 Sweden	Christina Berglund Order Administrator	0921-12 34 65 0921-12 34 67

file:///C:/tmpTransfer/TestHTML_0000001.GIF
Computer | Protected Mode: Off
100%

Figure 9. Several HTML attribute properties are identified in this HTML output.

Figure 10 shows the HTML.Anchor tag used to create a bookmark on the Customer field. Notice the URL entered into the address window ends with "#CENTC" which tells the browser to open the document and jump to the "CENTC" anchor tag.



file:///C:/tmpTransfer/TestHTML.HTM#CENTC - Windows Internet Explorer
file:///C:/tmpTransfer/TestHTML.HTM#CENTC
Live Search
TestHTML.HTM
Page
Tools
Computer | Protected Mode: Off
100%

Cust ID	Company Name & Address	Contact Name & Title	Phone & Fax
CENTC	Centro comercial Moctezuma Sierras de Granada 9993 México D.F. 05022 Mexico	Francisco Chang Marketing Manager	(5) 555-3392 (5) 555-7293
CHOPS	Chop-suey Chinese Hauptstr. 29 Bern 3012 Switzerland	Yang Wang Owner	0452-076545
COMMI	Comércio Mineiro Av. dos Lusíadas, 23 Sao Paulo SP 05432-043 Brazil	Pedro Afonso Sales Associate	(11) 555-7647

Figure 10. Use the HTML.Anchor attribute property to place bookmarks in the HTML output.

HTML.CSSFile, HTML.CSSClass.OverrideFRX and HTML.CSSClass.ExtendFRX attribute properties

When a report is output to HTML, the generated HTML includes styles embedded at the top of the document based on XSLT obtained from the GetDefaultUserXSLTAsString method in the HTMLListener class. A style class is defined for each object on the report using a naming convention of FRXn_r where "n" represents the report number and "r" represents the record number in the FRX. For example, if two reports are chained together, a style class named FRX1_7 relates to record number 7 from the first FRX and a style class named FRX2_12 relates to record number 12 from the second FRX. The following is a typical style class definition generated by Visual FoxPro.

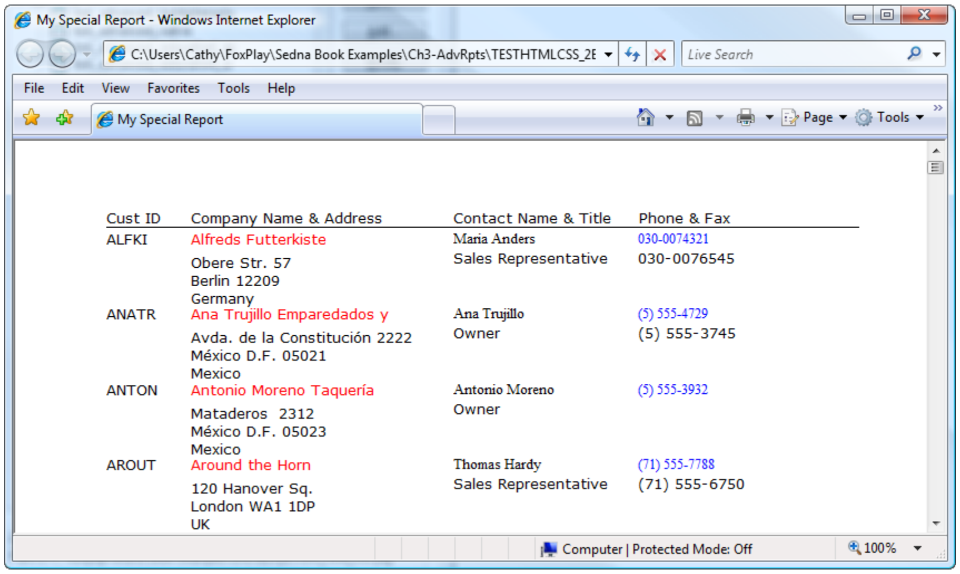
```
.FRX1_7{
  text-align:left;
  direction:ltr;
  vertical-align: top;
  font-family: "Verdana";
  font-size: 10pt;
  border: 0px none;
  padding: 0px;
  margin: 0px;
  font-weight: normal;
  color:#000000;
  background-color:transparent;
  overflow:hidden;
  position: absolute;
   }
```

You can use the HTML.CSSFile attribute property on the Document Properties tab to indicate the name of your own custom Cascading Style Sheet in which you have defined your own custom style classes. You can then use the HTML.CSSClass.OverrideFRX and HTML.CSSClass.ExtendFRX attribute properties on any given object on a report to tell Visual FoxPro to use the specified style class defined in the CSS indicated by the HTML.CSSFile attribute property.

Use the HTML.CSSClass.OverrideFRX attribute property to indicate the style class that should be used instead of the one created by Visual FoxPro. When using this attribute property, be sure your style class definition includes all the attributes normally included by Visual FoxPro or you may get undesired results. For example, if your style class only includes a font name and font color, text using that class shows up on the HTML using that font name and color, however, the position of the text is in the upper-left corner of the document instead of where it should be.

Use the HTML.CSSClass.ExtendFRX attribute property to indicate the style class that should be used in addition to the one created by Visual FoxPro, meaning both style classes are applied. The default Visual FoxPro style class is applied first, followed by your style class. This is important to remember because with Cascading Style Sheets, if both style classes include the same attribute property, the last one wins.

The output shown in **Figure 11** was generated using a Cascading Style Sheet to alter the color and font of some of the fields. The HTML.CSSFile attribute property of the report is set to "CPReport.css" (it's an expression so be sure to wrap the name in quotes). The Company Name and Contact Name objects have the HTML.CSSClass.ExtendFRX attribute property set to "CPChangeColor" and "CPChangeFont", respectively. The Phone Number object has the HTML.CSSClass.OverrideFRX attribute property set to "CPOverride".


My Special Report - Windows Internet Explorer
C:\Users\Cathy\FoxPlay\Sedna Book Examples\Ch3-AdvRpts\TESTHTMLCSS_2E
Live Search
File Edit View Favorites Tools Help
My Special Report
Page Tools
Computer | Protected Mode: Off
100%

```
Cust ID   Company Name & Address          Contact Name & Title    Phone & Fax
ALFKI     Alfreds Futterkiste             Maria Anders            030-0074321
          Obere Str. 57                   Sales Representative    030-0076545
          Berlin 12209
          Germany
ANATR     Ana Trujillo Emparedados y      Ana Trujillo            (5) 555-4729
          Avda. de la Constitución 2222   Owner                   (5) 555-3745
          México D.F. 05021
          Mexico
ANTON     Antonio Moreno Taquería         Antonio Moreno          (5) 555-3932
          Mataderos  2312                 Owner
          México D.F. 05023
          Mexico
AROUT     Around the Horn                 Thomas Hardy            (71) 555-7788
          120 Hanover Sq.                 Sales Representative    (71) 555-6750
          London WA1 1DP
          UK
```

Figure 11. Use the HTML-related attribute properties to alter the HTML output.

It’s important to note that when viewing the HTML, the CSS file indicated by the HTML.CSSFile attribute property must be in the same path as the HTML file if the full path name is not used in the HTML.CSSFile attribute property. The following is what is defined in the CPReport.css file used to generate the report shown in **Figure 11**.

```
.CPChangeColor {color: #ff0000;}
.CPChangeFont {font-family: "Times New Roman";}

.CPOverride{
  text-align:left;
  direction:ltr;
  vertical-align: top;
  font-family: "Times New Roman";
  font-size: 10pt;
  border: 0px none;
  padding: 0px;
  margin: 0px;
  font-weight: normal;
  color:#0000ff;
  background-color:transparent;
  overflow:hidden;
  position: absolute;
   }
```

HTML.Metatag.HTTP-EQUIV attribute property

The purpose of HTTP-EQUIV is to tell the browser to pretend an additional header was sent. The HTML.Metatag.HTTP-EQUIV attribute property lets you include one or more of these additional header properties in the generated HTML. The HTMLListener class expects the value of this attribute property to be XML and it’s quite picky about what’s in it. Unfortunately, the Visual FoxPro help file doesn’t give you any information at all about how to enter information into this attribute property.

If you want to enter multiple header properties, you can enter them like the following example. This tells the browser to redirect to a different page after 3 seconds. It also tells the browser not to cache the page.

```
<mydata>
  <meta name="Refresh" content="3;URL=http://www.frontier2000.com"/>
  <meta name="PRAGMA" content="NO-CACHE"/>
</mydata>
```

If you want to enter one single header property, you can shorten the format and enter it like this:

```
<meta name="Refresh" content="3;URL=http://www.frontier2000.com"/>
```

HTML.TextAreasOff attribute property

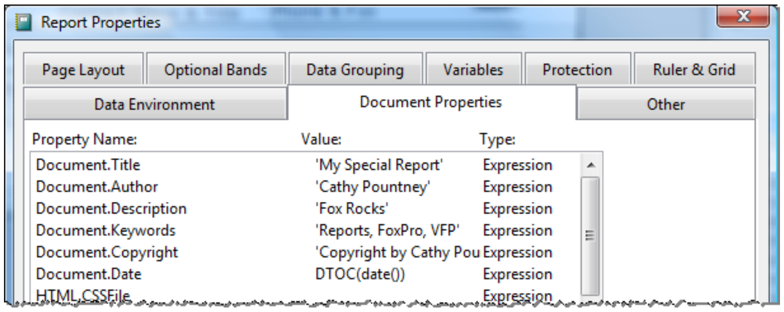
By default, the generated HTML includes the <TextArea> attribute for each of the text objects that overflow to additional lines on the report. You can set the HTML.TextAreasOff attribute property to Yes to omit the <TextArea> attribute throughout the report.

Document-related attribute properties

Several attribute properties on the Document Properties tab relate to some common attributes of HTML documents as well as many other types of documents such as Microsoft Word, Microsoft Excel, and PDF files. Use of these attribute properties requires the report to be run using the HTMLListener class in the _ReportListener class library in the FFC directory, or a class derived from the HTMLListener class.

- **Document.Title** (Expression)**:** The title of the document.
- **Document.Author** (Expression)**:** The author of the document.
- **Document.Description** (Expression)**:** The description of the document.
- **Document.Keywords** (Expression)**:** The keywords associated with this document.
- **Document.Copyright** (Expression)**:** The copyright notice associated with this document.
- **Document.Date** (Expression)**:** The date of this document.

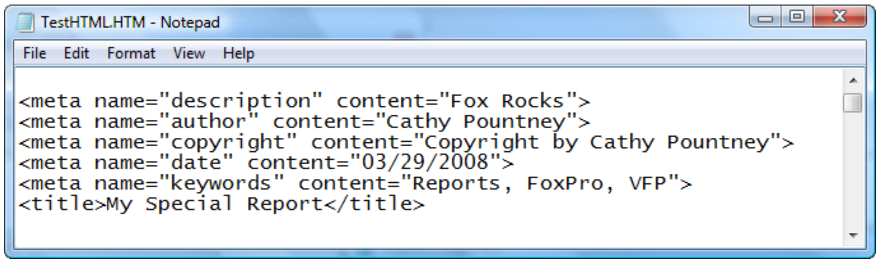
You can set the document-related attribute properties from the Document Properties dialog. The example shown in **Figure 12** sets the Title, Author, Description, Keywords, Copyright, and Date attribute properties. Notice the expression for the Date attribute property and the fact that the resulting value is a string. It's important to remember that each attribute property must evaluate to a string value so literal strings must be surrounded by quotes.



Property Name:	Value:	Type:
Document.Title	'My Special Report'	Expression
Document.Author	'Cathy Pountney'	Expression
Document.Description	'Fox Rocks'	Expression
Document.Keywords	'Reports, FoxPro, VFP'	Expression
Document.Copyright	'Copyright by Cathy Pou	Expression
Document.Date	DTOC(date())	Expression
HTML.CSSFile		Expression

Figure 12. Use the Document Properties tab to set several document-related tags in the HTML output.

Each of the above attribute properties relate to tags in the generated HTML output. **Figure 13** shows the HTML source generated when a report is run with the above attribute properties.



```
<meta name="description" content="Fox Rocks">
<meta name="author" content="Cathy Pountney">
<meta name="copyright" content="Copyright by Cathy Pountney">
<meta name="date" content="03/29/2008">
<meta name="keywords" content="Reports, FoxPro, VFP">
<title>My Special Report</title>
```

Figure 13. The HTML source reflects attribute properties entered on the Document Properties tab.

Distribution Issues

When packaging up your application for distribution, it's important to include a few things to ensure the new features run correctly on your client's system.

By default, VFP 9 ships with the SET REPORTBEHAVIOR setting at 80. This was done on purpose so all existing reports in your application continue to run as-is. Previous to VFP 9, GDI was used to render objects on reports. In VFP 9, GDI+ is used to render objects. GDI+ has better rendering, however, on some occasions things can render slightly larger than they did with GDI. This can cause a few problems. If a field isn't defined wide enough for the slightly extra width, you could see ******** in the output instead of the expected figures. On some occasions, I've even seen the height affected slightly and nothing appears to render for the object until you increase the height by one or two pixels.

To globally take advantage of all the new features in your application, SET REPORTBEHAVIOR 90 in the startup program for your application. However, be sure to test each and every report and make adjustments where needed for differences in rendering.

If you don't want to change the behavior throughout your entire application, you can issue the SET REPORTBEHAVIOR before running each report and change it back when done. Another option is to use the TYPE or OBJECT clause on the REPORT FORM command. That tells VFP to automatically assume 90 behavior for this one report.

When building your EXE, include the FFC files such as _FrxCursor, _GDIPlus, and _ReportListener.vcx in your EXE.

When distributing your application, be sure to distribute the three report-related app files:

- ReportBuilder.app
- ReportOutput.app
- ReportPreview.app

Setting REPORTBEHAVIOR, including the FCC files in your EXE, and distributing the three report-related app files ensures your application can take advantage of all the new reporting features.

Summary

In this session, we learned how to properly install VFP 9 SP 2. Then we learned all about the new Dynamics and Advanced features added to the product. With things like conditional formatting, resizing of objects, rotation of objects, and several HTML properties, you can now add a lot more pizzazz to your reports. Oh, and be sure to follow the build and distribution issues we've discussed too.

I encourage everyone to dive in and start using these new reporting features. You'll be happy, and more importantly, your customers will be dazzled with your new skills!

Biography

Cathy Pountney has been developing software for almost three decades and is proud to have earned the Microsoft Visual FoxPro MVP Award seven years in a row. She is equally proud to have had the opportunity to work as a subcontractor onsite in Redmond with the Microsoft Fox Team in 2001. Cathy enjoys writing articles for various Fox magazines as well as writing books. She authored The Visual FoxPro Report Writer: Pushing it to the Limit and Beyond and co-authored Visual FoxPro Best Practices for the Next Ten Years and Making Sense of Sedna and SP2. Cathy participates in her local FoxPro user group (GRAFUG) and speaks at other user groups when time permits. She has spoken at numerous conferences including GLGDW, Essential Fox, Advisor DevCon, DevTeach, and of course, her favorite, Southwest Fox. For the past several years, Cathy has worked for Memorial Business Systems writing software for the cemetery and funeral home industry, which proves ... Fox is NOT dead!

Email: *cathy@frontier2000.com*
My website: *www.frontier2000.com*
MBS's website: *www.mbs-intl.com*
Blog: *www.cathypountney.blogspot.com*
Twitter: *frontier2000*