New reporting features in VFP 9.0 SP2

Martin Haluza, Visual FoxPro DevCon, Prague, 2007

Table of Contents

Introduction

This session describes new features in the reporting system in Visual FoxPro 9.0 SP2. VFP 9.0 brought a significant extension to the reporting system by introducing XBase components cooperating with the basic engine during design time (REPORTBUILDER.APP) and run time (REPORTOUTPUT.APP, ReportListener class). The new features in SP2 focus on

- allowing users to extend the XBase components and define an interface mechanism between the design time and run time components (ReportBuilder vs. ReportListener) by exposing the report Memberdata structure in a standardized and easy to access way
- using this mechanism to implement most common extensions, such as Dynamic formatting

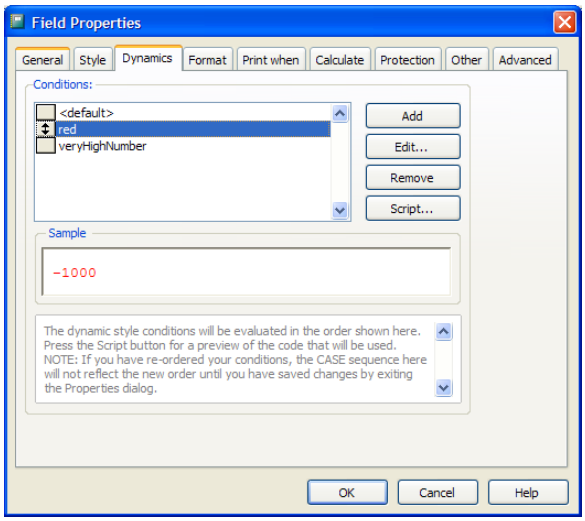
Downloading and installing VFP 9.0 SP2 Beta

This session is based on the SP2 beta version, which is available for download at http://msdn.microsoft.com/vfoxpro together with the Sedna beta. The service pack can be applied over VFP 9.0 or VFP 9.0 SP1. Should you have installed the SP2 CTP (Community technology preview) before, make sure you uninstall it before installing the SP2 beta.

New features – design time

Dynamic style

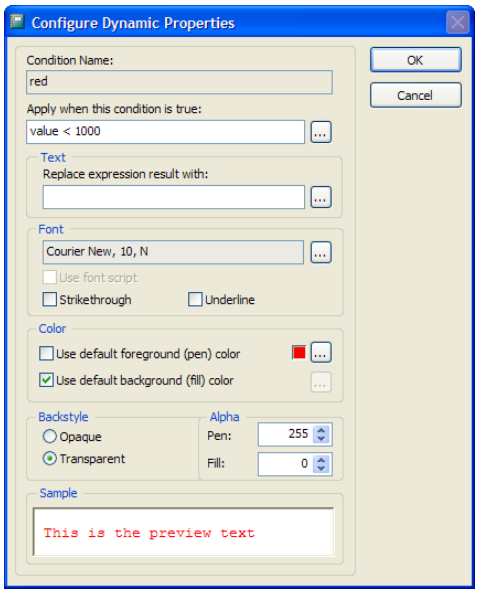
There is a new tab in the Field and Rectangle Properties – "Dynamics". It allows for adding multiple logical "Condition expressions" and style "Property modifications" to each field and rectangle. These conditions are evaluated, in a defined order, during the report processing, before the corresponding object renders. Once one of the conditions evaluates to .T. the defined modifications apply to the object.



The "Script…" button generates a FoxPro source code script that will be used in the report processing to perform the dynamic style evaluation. This script is not stored anywhere and is not intended to be modified, however, you can copy the code and use it as a run-time extension script (see the Custom scripts paragraph below).

Note: There are not separate scripts for each condition – all conditions are handled by a single script where each condition corresponds to a CASE clause in a DO CASE … ENDCASE statement. This means two conditions will never be applied at the same time. If two conditions would evaluate as *true* only the 1st one would be applied.

Following is the list of properties that can be modified:



- Field
  - Text
  - Font, font style
  - Color
  - Back style (opaque vs. transparent)
  - Alpha
- Rectangle
  - Width
  - Height

The dynamic properties actually correspond to parameters of EvaluateContents (for fields) and AdjustObjectSize (for rectangles) ReportListener methods in VFP 9.0 – so this in fact is a user interface to this functionality that allows programmers and end users to define dynamic properties without need to modify the report listener source code.

Example: Sales.prg, Sales2.frx, Sales3.frx – dynamic back style to highlight high and low number is a sales summary report

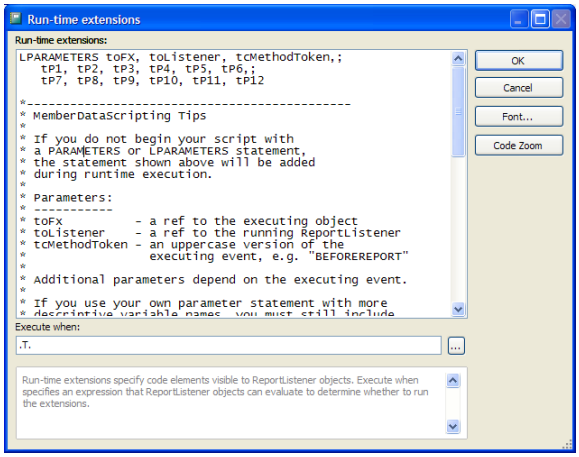
Implementation details

The Dynamic style properties are stored in the object's member data XML (STYLE column in the FRX) as type "R" <reportdata> section named as "Microsoft.VFP.Reporting.Builder.EvaluateContents". For example, the dynamic style "RED" in the screenshot is stored as:

```
<reportdata name="Microsoft.VFP.Reporting.Builder.EvaluateContents"
type="R" script="" execute="red" execwhen="value &lt; 1000" class=""
classlib="" declass="" declasslib="" penrgb="255" fillrgb="-1" pena="255"
filla="0" fname="Courier New" fsize="10" fstyle="0"/>
```

Custom scripts

The Run-time extensions box, in VFP 9.0 available in the Other tab of Label, Field, Line, Rectangle and Picture/Ole Bound Properties, is now also available in the Band Properties and Report properties.



While this dialog window was already available in VFP 9.0, it served as some kind of a text/code place holder available for programmers, without actually being looked at by the standard report listeners. In SP2 the scripts follows a specific interface and is evaluated by the default report listener. The extension window is no longer empty in SP2 – it contains a script skeleton with parameters description and some tips how to create the scripts. The parameters of the script follow the FXListener helper interface (see *New features – run time* paragraph below).

The "Execute when" field can contain one of the following values:

- An event name string (such as "RENDER"), case insensitive
- An expression evaluating to a logical value
- A "|" delimited list of event names (such as "RENDER|BEFOREREPORT")

The scripts are conditionally executed on runtime by the new FXListener report listener (see *New features – run* time below) if the "Execute when" expression evaluates to true (.T.) or the even name matches.

The following table list object types and events available:

	LoadReport / UnloadReport	BeforeReport / AfterReport	BeforeBand / AfterBand	Render	EvaluateContents	AdjustObjectSize
Report	√	√				
Band			√			
Label				√		
Field				√	√	
Line				√		
Rectangle				√		√
Picture				√		√

Implementation details

The custom script is stored in object's member data <reportdata> section type "R" with an empty name, "Execute when" field is stored in the *execwhen* attribute and actual script is stored in the *execute* attribute.

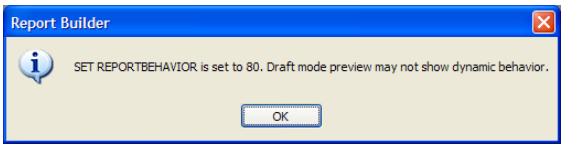
Example: Sales.prg, Sales4.frx – dynamic back style to highlight high and low number is a sales summary report

If the custom script modifies the field or rectangle properties in EvaluateContents or AdjustObjectSize and a dynamic style applies at the same time, the dynamic style properties will take precedence.

The run-time extensions dialog is not available in PROTECTED mode – it is intended to be used by programmers while end users could "fine-tune" the styles using the Dynamics tab features.

"Draft" previewer mode

While most new report builder features in VFP 9.0 were supposed both by the "old" and "new" reporting engine, the dynamic features in SP2, being implemented using report listeners, are supported by the "new" engine only. If the report builder runs with SET REPORTBEHAVIOR to 80 and you will try to previewer the report, you will get a message box informing about the dynamic features being switched off.



Advanced properties

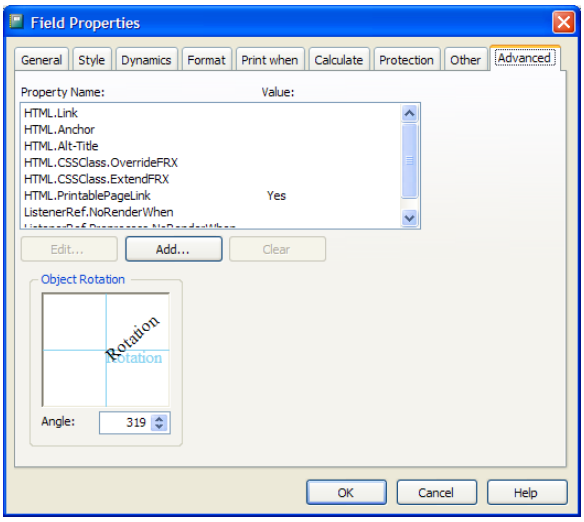
There is a new tab on report controls properties and report properties dialogs which contains a list of "Advanced Properties". Each property is identified by a unique name and can contain values of the following types:

- Expression
- Text/XML
- String
- File
- Yes or No

The list is populated with the default properties for the given object type (see implementation details below) and the user is allowed to modify the property values and add new properties.

The default properties included in the product include:

- HTML-specific properties for defining hyperlinks and CSS styles
- General document properties – such as document name, author, keywords, etc.
- ListenerRef.NoRenderWhen and ListenerRef.Preprocess.NoRenderWhen – conditional rendering – see HTMLSales example below



The tab also contains an object rotation control that allows for entering a rotation angle. These properties are made available for the ReportListener object when the report is processed.

Example: Sales.prg, Sales4.frx – labels rotation

Example: HTMLSales.prg, SalesSummary.frx – the sample shows how to define hyperlinks (both external and internal)

Example: HTMLSales.prg, HTMLSales.frx – performs a listener type conditional rendering

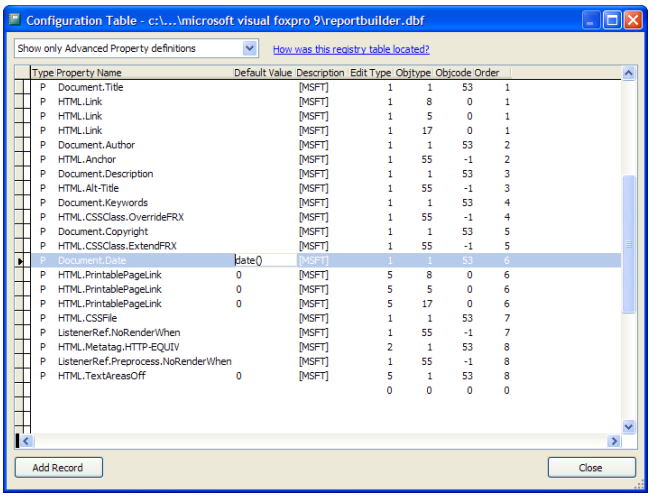
Example: Sales.prg, Sales5.frx – Decoration object triggered by an advanced property value – "Field.Class". The property value is an expression that defines the decoration object class.

Implementation details

The advanced properties are stored in object's member data <reportdata> section with "Microsoft.VFP.Reporting.Builder.AdvancedProperty" name, property name stored in the *execwhen* attribute and value stored in the *execute* attribute.

The rotation angle is stored in <reportdata> section with "Microsoft.VFP.Reporting.Builder.Rotate" name, the rotation angle stored in the *execute* attribute.

Default properties definition is stored in the report builder configuration table as type "P" records.



Extending the builder U/I

Another new feature in SP2 is the way to extend properties dialog boxes in the report builder – custom pages can be added to all properties dialogs as well as the "Multiple Selection" dialog. Each extension gets control when the dialog is about to show up (LoadFromFRX method) and when the [OK] button is clicked (SaveToFRX method). The extension can modify columns in the FRX file or the current control's memberdata, which are supplied via "memberdata" cursor.

There is a very simple interface that the extensions must follow:

- The extension must be derived from a Page class
- The controls on the page are contained in a container object
- The container object may contain two methods: LoadFromFRX and SaveToFRX. If these methods exist the builder will call them when appropriate.

The extensions are defined in the report builder configuration table as "T" type records. Event column is set to 1 – Properties dialog, ObjType controls the dialog type:

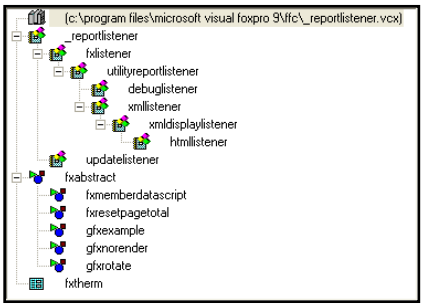
- 1 – Report/Global
- 5 – Label
- 6 – Line
- 7 – Rectangle
- 8 – Field
- 17 – Picture/OLE Bound
- 9 – Band
- 10 – Grouped Objects
- 99 – Multiple Selection
- 55 – Any Report control or layout element

Example: MyReportBuilder.dbf, MyRB.prg. MyReportBuilder.dbf, tabStretch class. A custom tab is defined allowing "stretch" feature for labels and fields. The stretch controls are implemented both for label and field control properties.

Example: MyReportBuilder.dbf, MyRB.prg. MyReportBuilder.dbf – a multi-select tabFieldClass tab allows for setting the "Field.Class" property value to multiple objects.

New features – run time

All FFC ReportListener classes are now descendants of the FXListener class. The update listener has been removed from the hierarchy as its functionality is now performed by FXListener itself with the help of fxtherm class helper.



FXListener class

The FXListener class contains two member collections of helper objects that get invoked during the report processing to perform extension tasks:

- FXS (helper objects that adjust the format and the content of the visual objects on the reports) and
- GFXS (helper objects that adjust or replace GDI+ rendering).

Each helper object must implement the following interface:

```
DEFINE CLASS HelperObject as CUSTOM

      Procedure ApplyFX
      LPARAMETER toListener, tcMethodToken, tp1, tp2, tp3, tp4, ;
      tp5, tp6, tp7, tp8, tp9, tp10, tp11, tp12

ENDDEFINE
```

Whenever an event is fired the FXListener goes through all helper objects in the collection and call their ApplyFX method, if appropriate, with the listener reference as the 1st parameter, event name as the 2nd parameter and the event parameters as the following tp1..tp12 parameters.

The return value of the ApplyFX method is ignored for FX objects. For GFX objects the return value of the Render method control how the default rendering will be performed:

Return value	Behavior
`OUTPUTFX_BASERENDER_RENDER_BEFORE_RESTORE`	The default rendering will be performed and then the stored GDI+ graphics state will be restored (for example, the helper object could have changed the coordinates)
`OUTPUTFX_BASERENDER_NORENDER`	The default rendering will be suppressed
`OUTPUTFX_BASERENDER_AFTERRESTORE`	The stored GDI+ graphics will be restored and then the default rendering will be performed
`(other values)`	The default rendering will be performed

The FXListener requires and automatically instantiates the following helper objects:

FXFeedback

Collection: FXS
Default location: fxTherm / _Reportlistener.vcx, stored in fxFeedbackClass, fxFeedbackClassLib and fxFeedbackModule properties of the FXListener class
Purpose: provides a visual progress bar during the report execution

MemberDataScript

Collection: FXS
Default location: fxMemberDataScript / _Reportlistener.vcx, stored in fxMemberDataScriptClass, fxMemberDataScriptClassLib and fxMemberDataScriptModule properties of the FXListener class
Purpose: evaluates the run-time extensions custom scripts

NoRender

Collection: GFXS
Default location: gfxNoRender / _Reportlistener.vcx, stored in gfxNoRenderClass, gfxNoRenderClassLib and gfxNoRenderModule properties of the FXListener class
Purpose: Evaluates “ListenerRef.NoRenderWhen” and “ListenerRef.Preprocess.NoRenderWhen” advanced properties to handle conditional rendering

Rotate

Collection: GFXS
Default location: gfxRotate / _Reportlistener.vcx, stored in gfxRotateClass, gfxRotateClassLib, and gfxRotateModule properties of the FXListener class.
Purpose: Performs object rotation

Adding custom helper objects

Custom helper objects can be added using the FXListener’s AddCollectionMember method. The method instantiates the object and add it to one of its member collections.

AddCollectionMember(tcClass, tcClassLib,tcModule,tlSingleton, tlInGFX, tlRequired)

tcClass
The object’s class name

tcClassLib
The location of object’s VCX library

tcModule
The object’s module

tlSingleton
If set to .F. repeated AddCollectionMember call will always be adding new instances of the specified class. If set to .T. only one instance of the class will be added (repeated calls will be ignored)

tlInGFX
If set the .T. the object will be added to the GFXs collection otherwise it will be added to the FXs collection

tlRequired

If set to .T. an error message box will be displayed if the object can't be instantiated

Return values

OUTPUTFX_ADDCOLLECTION_SUCCESS – the object was successfully instantiated and added to the collection
OUTPUTFX_ADDCOLLECTION_FAILURE – the object couldn't have been instantiated
OUTPUTFX_ADDCOLLECTION_UNSUITABLE – the object was instantiated but it doesn't implement the ApplyFX method

Example: FXEvents.prg – a simple example of a helper class that print event names to the screen

Example: Stretch.prg – a helper object that performs the stretched rendering using the tabStretch values from the "custom reportbuilder field properties tab" sample

New ReportListener base class properties

CallAdjustObjectSize, CallEvaluateContents

In VFP 9.0 the engine checks to see if there is any code in the AdjustObjectSize and CallEvaluateContents event methods of the lead listener and calls these methods during the report processing if it finds any. Listeners without dynamic behavior should have had these methods empty; otherwise a significant performance penalty occurred. However, the performance penalty still occurred for listeners allowing for dynamic behavior when no dynamic behavior actually took place.

In SP2, these two properties allow the listeners to indicate whether each of these events should be triggered.

Value	Result behavior
0 (default)	The VFP 9.0 behavior – the events are triggered if any code exists in the methods
1	The event will not be triggered
2	The event will always be triggered, even if there is no code. This option is needed if BindEvents is used for the event

Example: Sales.prg, Sales5.frx – decorator object instantiated by a helper object, controlled by an advanced property. It "manually" switches on the EvaluateContents method call as FXListener would not call it by default.

ReportListener.CommandClauses.File property

The ReportListener.CommandClauses.File property value is read-write in SP2. The native code will read this property after LoadReport and use it as the FRX file location. This change enables users to preprocess the FRX and save it as a private copy without changing the original FRX.

Example: NoRender helper object uses this property to create a temporary FRX with objects removed as defined by the ListenerRef.Preprocess.NoRenderWhen advanced property.