

## Reporting Efficiencies: What You Need to Know

By Art Greenhaus

Everyone wants requests to execute as efficiently as possible. After all, who has excess capacity on their machine, just waiting to be used, or extra time to waste while waiting for a request to finish?

WebFOCUS tries to be as efficient as possible when extracting data, but there are some things about it that users need to know. This is because when everything works, it doesn't matter, but when something goes wrong, it's usually at the worst time.

First, let's discuss, in general, what happens when a request is made. Any selections against the data that can be made, usually are. In most engines, this will limit the number of records returned. However, if a selection uses something that cannot be passed to the engine, all records may have to be retrieved. Most `DEFINE` expressions are passed.

An example of something that cannot be passed might be the selection on the result of a User Written Subroutine (UWS). Since the engine cannot evaluate the UWS, all records that pass any other selection must be returned. Then the expression can be evaluated, and a decision can be made as to whether the returned record is acceptable. This ensures that the minimum number of records is returned, which is our first goal.

Once records are retrieved, they are placed in the "internal matrix." The records are placed in the correct sort sequence (based on sort fields), and accumulated with other records of the same sort fields, if aggregation is requested. There is overhead in this process, so if the records are already in the desired sort sequence, not creating the internal matrix can also reduce resources used.

Finally, the results of the final internal matrix are used to produce the formatted output. Some formatting, such as HTML, can be very "wordy," increasing the amount of data returned by orders of magnitude. Minimizing the overhead of formatting makes it possible to gain further efficiencies.

With all that said, what can a user do?

For retrieval, test on real fields instead of a flag whenever possible. To test on whether an expression is true or not, the code can be used in a `WHERE` test. Thus, rather than testing like this:

```
DEFINE FILE xxx
TEST/A1 = IF expression THEN 'Y' ELSE 'N';
END
```

```
TABLE FILE xxx
```

```
WHERE TEST EQ 'Y'  
...
```

A simple:

```
TABLE FILE xxx  
WHERE expression  
...
```

Will accomplish the same thing. Many of these expressions are already decomposed and passed to the engine, so this may not save much. But for older releases, where the decomposition of expressions was less intelligent, it could have significant gains.

Another gain in record processing, especially against non-SQL engines, can be gained by letting WebFOCUS know that the records in the file are sorted. That way, once a position is found, beyond which the desired record can no longer exist, retrieval stops. This is like using a printed telephone directory (remember those?) to find a name.

Looking for a name beginning with *G*, I would never look at the letter *H* or beyond. This is controlled, in fixed-format files, by the setting `FIXRETRIEVE`, a value of `ON` respects sort order while `OFF` does not. You might wonder how WebFOCUS knows the sort order. It's determined by the `SEGTYPE` in the MFD.

As a word of caution, this can have a negative effect if you manually concatenate the data for a fixed-format file, and the records are not in sort sequence (i.e. putting `'ENGLAND'` after `'FRANCE'`), and you select on the sort key(s). Since the MFD indicates the records are sorted, the retrieval may stop before the desired record is found. By turning `FIXRETRIEVE` to `OFF`, the file search will always proceed to the end of the file.

With respect to the internal matrix, using the command `TABLEF` rather than `TABLE`, indicates that the internal matrix is not to be built, and output is to be produced with the records in the same sequence as the input. If the records are returned in the desired order, this can produce significant savings, especially with large answer sets.

For SQL retrieval, this is controlled by the setting `SQLTOPTTF`, which when set to `ON` (the default) will generate a `TABLEF`, if possible, rather than a `TABLE` command. There are some operations, such as `ACROSS`, which are not supported with `TABLEF`, and that is why the setting takes effect if possible, not always.

Lastly, if you can minimize the overhead of formatting in your output, you can significantly cut down the size of the returned output. HTML is especially verbose, with the default of having formatting tags for each table cell. If you use the setting `HTMLCSS`, and set it to `ON`, an internal Cascading Style Sheet will be generated, allowing formatting options for like-formatted cells to be reused.

# Introducing Guided Report Forms

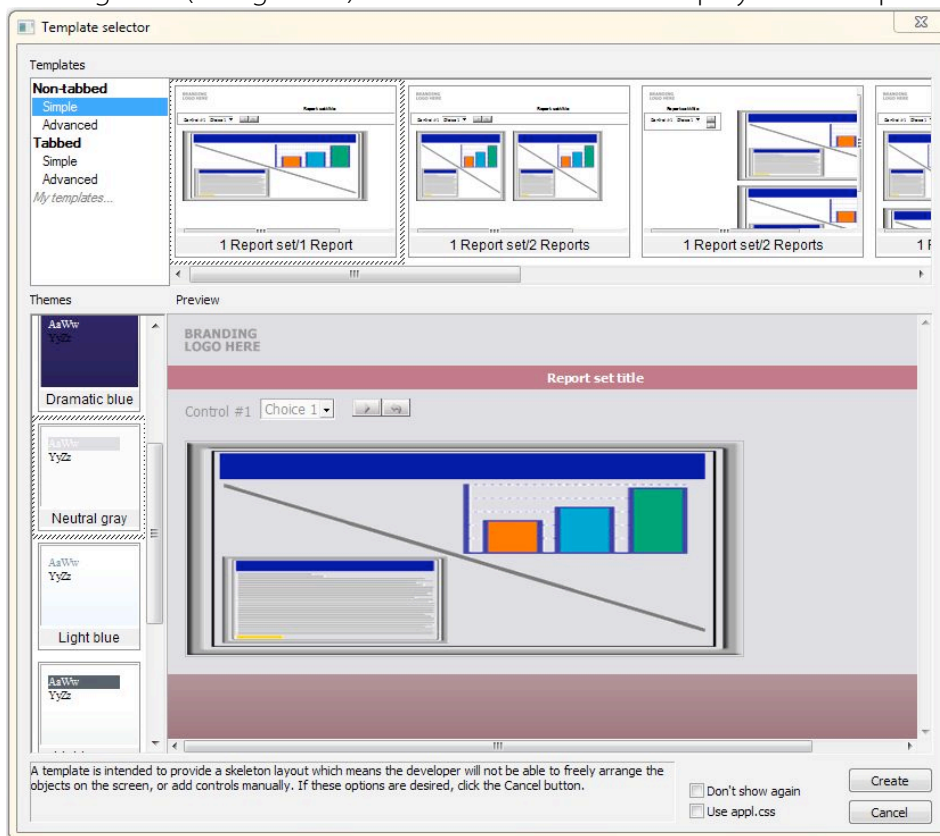
By Lisa Scipio

New in Developer Studio release 7.7.02, Report Painter promotes the authoring of reports with parameterized fieldnames, field options and data values. For optimum functionality, these reports should be embedded in a Guided Report Form, which is also new in release 7.7.02.

In the Guided Report Form, controls are arranged in an order that illustrates the relationship between their values, particularly fields and field options. Guided Report Forms further enhance the presentation of reports and charts by offering an array of templates and themes as well as a choice between tabbed and non-tabbed report sets.

Features involved in creating Guided Report Forms include the Template Selector, options in the Report Painter, such as Generate Parameter group, Field Options variables, Guided Drill Down and various additions to the HTML Composer that facilitate customization of the Guided Report Form.

When creating a new Guided Report Form, the developer is first presented with the Template Selector. However, this is not unique to the Guided Report Form since it is also available when creating new (non-guided) HTML Forms. **Screen 1** displays the Template Selector.



**Screen 1**

In addition to offering the ability to choose templates and themes, the Template Selector provides the option to apply **appl.css** to templates. When **Use appl.css** is selected, this file is copied from **ibi\DevStudio(n)\bin** to the first application in the **App** path. This

CSS file may be customized before or after it is copied to the app path and when in use, it applies styling overrides such as standard company colors or a company logo.

Themes enhance the templates, which may be further enriched by enabling complementary report or graph style files available in the report or graph tools. The following chart shows a list of themes and matching style files.

Template Theme	Report/Chart Style File
Yellow red	Redbronze
Beige blue	Golden
Dramatic blue	Blackbluepurple
Neutral gray	Graylight
Light blue	Bluemedium
Light green	Greengray
Light purple	Purplelight

“Guided Report Mode” is enabled in Guided Report Forms and incorporates the following:

- Prompt to select a data file only once per report set. The file selected to create the first report or graph is used for subsequent reports or graphs. Forms with multiple report sets may use different data files in each report set.
- **JOIN, DEFINE, WHERE, SET** statements as well as OLAP Dimension Builder clauses in the first component are copied to other embedded reports and charts in the report set.
- The Procedure Viewer option is not available on the right-click context menu. It is now possible to invoke the SET, JOIN, Dimension Builder and Define tools from the Report Painter, opened in the HTML Composer.
- The Document Composer option is not available in the right-click context menu.
- “Insert new report or graph before” and “Insert new report or graph after” options have been added to the right-click menu, which allow the insertion of additional frames to a Report Set.

Guided Report Mode is also available in the generic HTML Composer. To open a generic display of the HTML Composer in “Guided Report Mode,” select New->Guided Report Form, then cancel the Template Selector.

A RIA theme or Theme may be enabled at this stage; when one is active, the other option is removed from the Properties Window. The theme option is available only before objects are inserted on the canvas and so must be selected before any work begins. On the generic canvas in guided mode, a tab control functions like a tabbed report set offered by the Template Selector, and that means only one data file can be used on each tab.

In Managed Reporting, the “Save Selection” button is embedded in the templates. In addition, “Schedule” and “Defer” buttons are added to the Guided Report Form when procedures are referenced.

Options for altering the presentation of objects on the template include number of columns, display and label location. The number of columns may be set in the Properties Window to alter the alignment of controls on the Guided Report Form. By changing the Display value to "Do not display," unused objects are rendered invisible. Control labels may be moved "To the left of the input" by setting this in the Form Settings dialog or in the Properties Window.

In a future issue of WebFOCUS Newsletter, we will focus on embedding Guided Reports in the Guided Report Form.

## **Data Security: Limiting Access Based on Field Values**

By Noreen Redden

Security is a hot topic these days, and wiser people than I are addressing server security in WEBFOCUS. Access to datasets may be limited by various security protocols, such as RACF on the IBM Mainframe. However, FOCUS and WEBFOCUS go a step further, allowing the DBA to limit access based on field values.

WebFOCUS 7.7 contains two features that improve DBA, so let's review them now. Since the vast majority of us use WEBFOCUS for reporting, that's the emphasis in this article. However, remember that if FOCUS is used for MODIFY or MAINTAIN, DBA WRITE security may be specified

A quick review: DBA security is specified at the end of the MFD, following all field declarations, and any DEFINE, FILTER, and COMPUTE declarations. The word END begins the DBA section, which is followed by the declaration of the DBA password, and then the user declarations. If you need information on the basics of DBA, the documentation can be reviewed at this URL: [http://documentation.informationbuilders.com/masterindex/html/html\\_wf\\_761/wf761sec/index.htm?url=topic77.htm](http://documentation.informationbuilders.com/masterindex/html/html_wf_761/wf761sec/index.htm?url=topic77.htm)

Remember that information in the DBA section of the MFD is applied to any type of data source, so long as the retrieval is through FOCUS. User passwords can be from one to 64 characters and contain special characters. They are not case sensitive unless SET DBACSENSITIV = ON has been set. The default password, if never set, is blank, and is sometimes used to establish general restrictions for all users, who therefore do not set their passwords.

The PERMPASS parameter establishes a user password that remains in effect throughout a session or connection. You can issue this setting in any supported profile but it is most useful when established for an individual user by setting it in a user profile. It cannot be set in an ON TABLE phrase. It is recommended that it not be set in EDASPROF because it would then apply to all users, unless that is what is desired.

The user password may restrict access to specific segments (RESTRICT=SEGMENT, NAME=segment name) and specific fields (RESTRICT=FIELD, NAME = fieldname); allow a field to be specified in a request, but not return data base information (RESTRICT=NOPRINT, NAME=fieldname); or specify a selection test to be applied to all requests referencing a segment (RESTRICT=VALUE, NAME=segment name ,

VALUE= field relation value) or  
RESTRICT=VALUE\_WHERE, NAME=segmentname, VALUE= any acceptable WHERE  
test;). To globally apply a selection test, in all cases, use NAME=SYSTEM, rather than a  
segname. VALUE\_WHERE is a new feature in release 7.7.2, and allows expressions, ANDs and OR  
as needed.

**Example:** A data source contains userid, jobcode, managerid, and  
vacationdaysallowed. Manager1 may see vacationdaysallowed for himself and  
for anyone who reports to him. Employee1 may see vacationdaysallowed for himself  
only.

```
FILE=EMPDATA, SUFFIX=...
      SEGMENT/FIELD Definitions
END
DBA=DBA, $
USER=Employee1, ACCESS=R, RESTRICT=VALUE, NAME=SEG1, USERID EQ '123456789', $
USERID=Manager1, ACCESS=R, RESTRICT=VALUE_WHERE, NAME=SEG1, VALUE=USERID EQ
'176891239' OR MANAGERID EQ '176891239';, $
```

Now, problems arise when multiple data sources are involved. File1, which may or may not  
contain restrictions on access is JOINed to File2. (Either a dynamic JOIN or cluster JOIN in  
the MFD). Until 7.7.02, in a JOIN situation, the restrictions specified in the HOST file were in  
effect, and any restrictions in the guest files were ignored, assuming each file had its own  
restrictions specified in its specific MFD.

You cannot JOIN from an unrestricted file to a restricted SUFFIX=FOC or SUFFIX=XFOC file,  
because in those data sources, the DBA password is actually stored in the database. No other user  
restrictions are applied and other data sources do allow joining from an unrestricted file to a  
restricted one, which might cause a hole in security.

The EMPDATA FILE, containing EMPID, LASTNAME, and FIRSTNAME is not restricted.  
However, SALHIST does not allow access to the salary history for EMPID 1 for normal users.

```
FILE=EMPDATA, SUFFIX=...
      SEGMENT/FIELD Definitions
END
DBA=DBA, $
USER=USER1, ACCESS=R, $
```

```
FILE=SALHIST, SUFFIX=...
      SEGMENT/FIELD Definitions
END
DBA=DBA, $
USER=USER1, ACCESS=R, RESTRICT=VALUE, NAME=SEG1,
      VALUE=EMPID NE 1, $
```

However, a JOIN of the files and a subsequent TABLE would allow access since only the restrictions in EMPDATA would apply.

So, DBAFILE was born. With DBAFILE, each file simply specifies DBA=dbapassword, DBAFILE=mastername, \$ where the mastername specified contains all restrictions for all files. So, if EMPDATA joins to SALHIST, a possible implementation would be as follows:

```
EMPDATA:
FILE=EMPDATA, SUFFIX=...
    SEGMENT/FIELD Definitions
END
DBA=DBA, DBAFILE=EMPCTL, $

SALHIST:
FILE=SALHIST, SUFFIX=...
    SEGMENT/FIELD Definitions
END
DBA=DBA, DBAFILE=EMPCTL, $

amd EMPCTL
FILE=EMPCTL, SUFFIX=FIX
SEGNAME=ONE, SEGTYPE=S0
    FIELD=FIELD1,, A1, A1, $
END
DBA=DBA, DBAFILE=EMPCTL, $
USER=USER1, ACCESS=R, $
USER=USER2, ACCESS=R, $
FILE=SALHIST
USER=USER1, ACCESS=R, RESTRICT=VALUE, NAME=SEG1,
    VALUE=EMPID NE 1, $
```

**Note:** This is one case where the FILENAME= in the MFD must match the actual filename.

User USER1 has access to both files, but if his request accesses fields from SALHIST, the test IF EMPID NE 1 will be applied. Note that because VALUE is an "IF" test; quotes are not required around the value unless it contains an embedded blank.

That's great, but what happens if a HOLD file was created with no restrictions? We are back in the same problem, unless DBA information is automatically embedded in any HOLD files. Although it is possible to enforce that, with SET HOLDSTAT = filename, where information (including DBA info) provided in the specified file will be included in HOLD Masters. If that wasn't done, then the following series of requests could result in a security lapse.

```
SET PASS=USER1
TABLE FILE EMPDATA
PRINT EMPID
ON TABLE HOLD AS HOLD1
END
-RUN
JOIN EMPID IN HOLD1 TO EMPID IN SALHIST AS AJ
TABLE FILE HOLD1
PRINT *
```

```
IF USERID EQ 1
END
```

So we have a new feature documented in 7.7.02: SET DBASOURCE = HOST/ALL. DBASOURCE=HOST (the default) is the same as current DBA application. The access restrictions are enforced only from the host file in a JOIN structure or the last file in a COMBINE structure unless a DBAFILE is used to enforce access restrictions to other files in the structure.

DBASOURCE=ALL, on the other hand, requires the user to have read access to every referenced file in a JOIN or COMBINE structure.

So, in the prior example, trying to print information from SALHIST would give no data because the DBA restrictions USERID NE 1 would be applied, in addition to the WHERE condition coded in the request. Whether DBAFILE is specified and whether the host file is restricted, the specified USER must have access to all files being retrieved, and all access restrictions will be applied.

Now, we have had a few customers who wanted to be able to build the restrictions "on the fly" to make them more dynamic. There, we established a FOCUS file that stored the restrictions and whenever they changed, a simple TABLE request built the DBAFILE dynamically. I chose to store those restrictions in a FOCUS file, as it is accessible only by FOCUS/WEBFOCUS, and the data in a FOCUS file can itself be encrypted, which means it cannot easily be read by any other means. However, of course, it could be stored in any convenient datasource.

Here is the one version of security.mas :

```
FILE = SEC, SUFFIX=FOC, $
SEGNAME=ONE, SEGTYPE=S1
  FIELD=ADMINISTRATOR, DBA, A64, $
SEGNAME=ACSSEG, SEGTYPE=S3, ENCRYPT=ON, $
  FIELD=USERID,, A8, $
  FIELD=TABLENAME, ,A64, $
  FIELD=LISTNO,, I4, $
  FIELD=PSS,, A64, $
  FIELD=ACS,, A2, ACCEPT=R W RW U, $
SEGNAME=RESTSEG, SEGTYPE=S2, ENCRYPT=ON, PARENT=ACSSEG, $
  FIELD=RSTRCT,,A11,ACCEPT=FIELDNAME SEGNAME SAME VALUE VALUE_WHERE, $
  FIELD=RSTNO,, I4, $
  FIELD=RSTNAME,,A64,$
SEGNAME=VALSEG, SEGTYPE=S1, ENCRYPT=ON, PARENT=RESTSEG, $
  FIELD=VALNO,, I4, $
  FIELD=VALLNE,,A80,$
END
DBA=DBA, $
```

And the creation of the DBAFILE, SEC.FEX:

```
SET PAGE=NOPAGE
SET ALL = ON
SET PASS=DBA
FILEDEF SECOUT DISK ... SECOUT.MAS
TABLE FILE SEC
PRINT VALLNE NOPRINT
```

```

BY ADMINISTRATOR NOPRINT BY TABLENAME NOPRINT BY PSS NOPRINT
  BY ACS NOPRINT BY RSTRCT NOPRINT BY RSTNAME NOPRINT
  BY VALNO NOPRINT
ON ADMINISTRATOR SUBHEAD
"FILE=SECOUT,SUFFIX=FOC"
"SEGNAME=SEG1,SEGTYPE=S0"
" FIELD=DUMMY,,A1,"
"END"
"DBA=<ADMINISTRATOR , DBAFILE = SECOUT , $ "
ON TABLENAME SUBHEAD
"FILENAME = <TABLENAME , $ "
ON PSS SUBHEAD
"USER=<PSS , ACCESS = <ACS , "
WHEN RSTRCT NE ' ';
ON PSS SUBHEAD
"USER=<PSS , ACCESS = <ACS , $ "
WHEN RSTRCT EQ ' ';
ON RSTRCT SUBHEAD
"  RESTRICT = <RSTRCT , NAME=<RSTNAME , "
WHEN RSTRCT NE ' ' AND VALNO GT 0 AND VALNO NE LAST VALNO;
ON RSTRCT SUBHEAD
"  RESTRICT = <RSTRCT , NAME=<RSTNAME , $ "
WHEN RSTRCT NE ' ' AND VALNO LE 0;
ON VALNO SUBHEAD
"VALUE_WHERE = <VALLNE ;,$ "
WHEN VALLNE NE ' '
ON TABLE HOLD AS SECOUT FORMAT DOC
END

```

All of the files in the system have DBAFILE=SECOUT, which can be changed easily by a MAINTAIN (or even a MODIFY where the fixed format transaction is fed by an HTML form). Of course, this could be created for each user upon their connection, and &USERID may in fact be a system variable that identifies the user, without specific input.

If all restrictions are based on values, then another option is a static DBA, but dynamic value list:

```

FILE=securefile,SUFFIX=...
...
END
DBA=dbapassword,$
USER=userpassword,ACCESS=R,RESTRICT=VALUE_WHERE,NAME=SYSTEM,
  VALUE=EMPID IN FILE filelist;,$

SEC.FEX:
SET PASS=DBA
FILEDEF SECOUT DISK ... SECOUT.TXT
TABLE FILE SEC
PRINT VALS ON TABLE SAVE AS SECOUT
IF USERID EQ &USER
END

```

The only problem with either of the above approaches, was that, for some environments, it was not dynamic enough. The procedure to create the DBAFILE or IN FILE list was run in the EDASPROF, and restrictions might change at any time in a session and should be instantly applied.

So, in 7.7.2, there is a Candidate for Release (A new feature for which we need your input), which would allow the original MASTER to specify

```
FILE=securefile, SUFFIX=filetype, MFD_PROFILE=SEC , $  
...  
END  
DBA=dbapassword, DBAFILE=SECOUT, $
```

Or

```
DBA=dbapassword, $  
USER=userpass, RESTRICT=VALUE_WHERE ,  
VALUE=EMPID IN FILE SECOUT; , NAME=SYSTEM, $
```

The SEC.FEX will be executed with any TABLE / TABLEF / MATCH FILE/ GRAPH request.

Have fun, and be safe out there.

## Bye-Bye MR Applets

By Susan Trommer

As of WebFOCUS 7.6.5, the WebFOCUS Business Intelligence Products Division began to stabilize the Managed Reporting (MR) Applet user interface. That is when we established a plan to incrementally implement the administration and development capabilities available in the MR Applet user interface in the Business Intelligence Dashboard. This plan has been a priority project not only to consolidate the interfaces for Managed Reporting but to identify what release to discontinue support – not distribute – the MR Applet User Interface.

WebFOCUS 8 is the release in which support will be discontinued for the MR Applets because of the significant implementation changes needed to all MR user interfaces and report development tools to integrate with the new WebFOCUS Client Security Mode. In addition, Release 8 will launch the new BI Portal interface for Managed Reporting.

For those of you still using the MR Applet User Interface, it is important to understand that both the MR Applet user interface and BI Dashboard utilize the same Managed Reporting repository for storing application content and user security information. The value added with Dashboard is the ability to create a customized user interface to access Managed Reporting, including the ability to allow user personalization. BI Dashboard is a Section 508 Accessibility-compliant user interface. In addition, there are many new features implemented in BI Dashboard and Developer Studio's Managed Reporting Component in Release 7.6.9 and later releases that are not available in the MR Applet interface.

Following is an outline of the incremental introduction of development capabilities within BI Dashboard and the core Managed Reporting new features that are *not* available in the MR Applet user interface. See the Summary of New Features and Managed Reporting product documentation for complete information on the functionality listed.

(DC) – development capability

(NF) – new feature

### Release 7.6.5

- (DC) Ability to configure BI Dashboard to allow MR Administrators and Developers to create Standard Reports.

### Release 7.6.7

- (NF) Section 508 Accessibility – the BI Dashboard logon page has an accessibility link, as well as an accessibility link on the Dashboard banner that allows users to enable or disable accessibility.
- (NF) Cut, copy and paste custom reports within and across Custom Reports folders.

### Release 7.6.9

- (NF) New Domain property, Do not show on User's list, provides the ability to hide the domain from the end user display in the BI Dashboard.
- (DC) A full WebFOCUS Client installation by default configures BI Dashboard to allow MR Administrators and Developers to create Reporting Objects and Other Files. If you upgrade from Release 7.6.1 or later to Release 7.6.9 using the WebFOCUS Client Service Pack installation, and you want to use this new feature, see the Release 7.6 WebFOCUS Upgrade Considerations.
- (NF) Managed Reporting FEX and HTM File Properties options include the following new properties that restrict the usage of a report:
  - *Restrict Developer Access* property is only visible to, and can only be set by, a MR Administrator. When this property is set, the Managed Reporting Developer role capabilities are restricted to be at an end user level (Analytical Users and lesser roles). When this option is set, only the Administrator can manage the file (edit, delete, cut, copy, paste) and set property options.
  - *Schedule Only* property applies only to Standard Reports procedures (FEXs) and can be set by Managed Reporting Administrators, and by Developers when the Restrict Developer Access property is not enabled. When this property is set, the Standard Report can only be scheduled using ReportCaster. The Schedule property is applicable to end users (Analytical Users and lesser roles) and also to Developers when the Restrict Developer Access is also enabled.
- (NF) Ability to create Reporting Objects with InfoAssist. When InfoAssist license is configured in your WF Client configuration, the New Reporting Object dialog box provides the option to create a Reporting Object using InfoAssist.

### Release 7.6.10

- (NF) Ability to control reporting tool access at the global level or the view level in BI Dashboard. The configuration file, *bid-config.xml*, contains the variables that control whether to make a reporting tool available globally. The default setting for each tool variable is true. The reporting tools are the following:
  - Report Assistant
  - Graph Assistant
  - Advanced Graph Assistant
  - InfoAssist

- Power Painter
- Text Editor

### Release 7.6.11

- (NF) Users can cancel active requests that are initiated from their browser session on all available reporting servers by clicking *Stop Requests* under the Utilities menu on BI Dashboard's top banner.

### Release 7.7.01

- (NF) Ability to select the following property options when using the Import File Utility to add Standard Reports and Other Files to a Managed Reporting domain:
  - Prompt for Parameters (selected by default)
  - Run with OLAP
  - Only Run as Deferred Report
  - Schedule Only
  - Do not show on User's list (selected by default)
  - Restrict Developer Access
  - Show as Predefined Style
- The Standard Report property *Show on User's List* is changed to *Do not show on User's list* when creating a new Standard Report in Release 7.7, the *Do not show on User's list* property is selected by default, which makes the report unavailable to end users when it is created. You can change the *Do not show on User's list* property before saving the report in those save dialog boxes that include this property, or after it is created by right-clicking the report and selecting the *Properties* option.
- (NF) The View Builder ReportCaster *Scheduling Tool Template* setting provides the ability to define the template that will be used with the new ReportCaster Scheduling tool in a Public or Group view. You can select the Standard or a Custom template to determine which options are available to the end user.

### Release 7.7.02

- (NF) MR Administrators and MR Group Authorization Managers can prevent users from saving, accessing, or scheduling My Reports in specific domains using the Restrict My Reports domain property. When a domain is restricted, all users, including MR Administrators, are no longer authorized to save, access, or schedule My Reports in that domain.

If you choose to utilize features listed above that use property settings that are only available from the BI Dashboard environment, your users should not use the MR Applet Interface because they will not have the same capabilities and may encounter unexpected results. If you would like to remove access to the MR Applet user interface from your WebFOCUS installation, please contact Customer Support Services.

If you have any questions or concerns regarding support for the MR Applet user interface being discontinued, please send me your feedback ([susan\\_trommer@ibi.com](mailto:susan_trommer@ibi.com)) or open a case with Customer Support Services.

## Enhancing the New JavaScript Grid With HTML

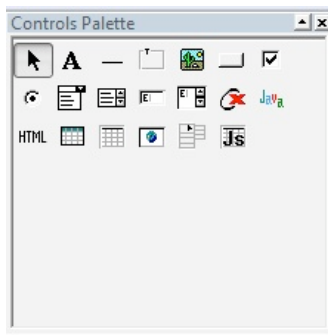
By Mark Derwin

In release 7.7 of WebFOCUS, a new control was added to the Maintain Development Environment (MDE). The JavaScript grid allows a user to display and update multiple rows and columns of data on a Maintain Winform. While we already had an Active-X grid, this object has several advantages.

The first advantage is that you no longer have to have the Active-X control installed on all computers displaying the object. This could sometimes be a problem if computers were not allowed to download objects because of security. In addition, you don't have to worry about the version of the object.

Second, the JavaScript grid can easily be manipulated with JavaScript commands. Third, you can use the JavaScript grid on browsers like Firefox and Safari. The Active-X grid could only be displayed on Internet Explorer. Lastly, you can easily add both images and HTML objects to grid cells for display and selections. This article discusses how to use HTML. Please note the Active-X grid is still available and fully supported.

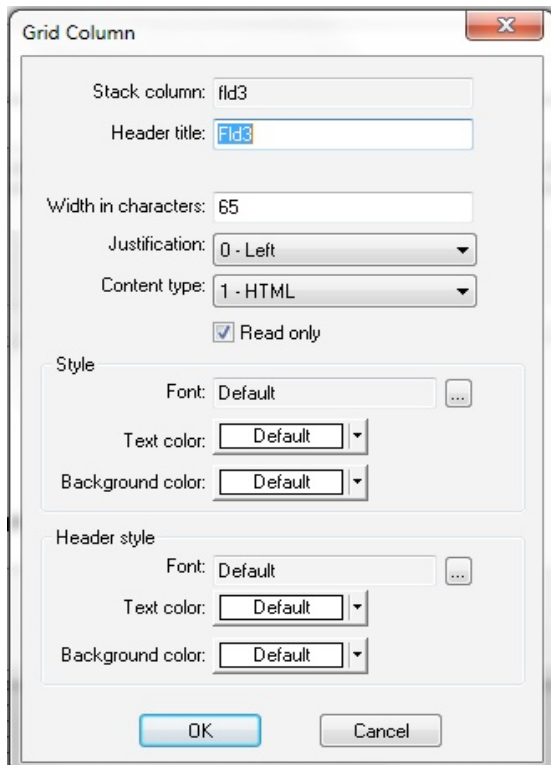
When creating a winform in the MDE, you can see the new object on the Controls Palette (**Screen 1**). The new grid displays the letters Js. You place it on the form and assign the source stack and fields the same way you would the old grid and the HTML Table objects.



**Screen 1**

When you double-click on a selected field, the Grid Column properties dialogue is displayed (**Screen 2**). In release 7.702, we have added the content type control that lets you tell Maintain what is being displayed. You can choose 0 – Text, 1 – HTML or 2 – ImagePath. If you want to display a Radio Group, set the content to HTML. You can change the display width of the object to 65 since the displayed value is much smaller than the format of the field. Also check the "Read only" checkbox. This allows the user to make a selection from the control, but not display the HTML code that populates the cell.

Now that you have told the grid to expect HTML in the cell, you have to create the stack to contain it (**Screen 4**). I am using the movies sample file in my example. I load five records into a stack name, **stk**. I then create my Radio Group in my field, **fld3**. Please note that while the display length is 65, the actual length of the field is an A250.



Make sure you allow enough space for all the selections and HTML code. Also note that I have to create this object in a loop since I want the Radio Group to appear on all rows of the grid. If I didn't loop through, it would only appear on the first row. Doing this also creates a unique name for each displayed Radio Group. This is needed for retrieval.

For this technique to work on all browsers, there is an event that we need to use. Bring up the events screen for the JavaScript grid and add the code you see on **Screen 3**. Make sure the form and grid names reflect what you have in your application. Also, since this is JavaScript code, remember that case matters.

**Screen 2**

```
function OnGrid1_OnClicked ( int col, long row, boolean updn, boolean processed) {
Form1.Grid1.setFocusedCell(null,0,false);
}
```

### Screen 3

If there are other actions you want to do on left-click, just add this line of code as well. You can also check the current column and only use it on each and every column that has embedded HTML.

Once the user has made their selections, you need to retrieve them. I added a button to my form that performs the case **tellme**. Here, I create a field in my stack called sel2. Use the **GetHTMLField** subroutine to get all the selected values from the Radio Group. For this example I am displaying them on my form in another grid (**Screen 5**).

```

MAINTAIN FILE movies
$$Declarations
Case Top
for 5 next moviecode into stk
compute i=1;
repeat stk.foccount
compute stk(i).fld3/a250 = "<input type=radio name=radio " || i ||
" value='POOR' CHECKED>POOR <input type=radio name=radio " || i ||
" value='FAIR'>FAIR <input type=radio name=radio " || i ||
" value='GOOD'>GOOD <input type=radio name=radio " || i ||
" value='GREAT '>GREAT"
compute i=i+1;
endrepeat
Winform Show Form1;
EndCase

case tellme
compute i/i2=1;
repeat stk.foccount
compute stk(i).sel2/a5 = Form1.GetHTMLField('radio' || I);
compute i=i+1;
endrepeat
endcase
END

```

## Screen 4

Moviecode	Fld3
1 001MCA	<input type="radio"/> POOR <input checked="" type="radio"/> FAIR <input type="radio"/> GOOD <input type="radio"/> GREAT
2 005WAR	<input type="radio"/> POOR <input type="radio"/> FAIR <input checked="" type="radio"/> GOOD <input type="radio"/> GREAT
3 020TUR	<input type="radio"/> POOR <input type="radio"/> FAIR <input type="radio"/> GOOD <input checked="" type="radio"/> GREAT
4 024WAR	<input type="radio"/> POOR <input type="radio"/> FAIR <input type="radio"/> GOOD <input checked="" type="radio"/> GREAT
5 031KKV	<input type="radio"/> POOR <input type="radio"/> FAIR <input checked="" type="radio"/> GOOD <input type="radio"/> GREAT

Sel2
1 FAIR
2 GOOD
3 GREAT
4 GREAT
5 GOOD

## Screen 5

The new JavaScript grid makes it easy for developers to create robust, and easy-to-use applications that can run on multiple browsers. Not only can users enter multiple rows and columns of data, they can make selections as well. While this example uses Radio Groups, you can just as easily use check boxes and drop down boxes.

## Embedded Analytics With Active Technologies API

By Yoshiko Akai

Business costs continue to grow as companies buy more hardware and software required for the generation and distribution of an increasing number of reports to a growing number of users. With this in mind, Information Builders has developed the Active Technologies API to specifically meet the demand for online information retrieval resulting from the ever-growing number of users and data volumes.

In a traditional report, the user views static data arranged and summarized for a single purpose. This is the primary reason many companies end up generating hundreds of reports in an attempt to satisfy the need of every user to view the same information in many different ways.

Active reports have an embedded interactive analytics engine to perform most user-requested functions, such as filtering, sorting and charting. When an active report with the built-in analytical

capability is presented, users can modify the reports and create their own versions to suit their needs. You can provide end users with the columns and rows needed to perform their own analytics in an active report so they can create and apply their own filters; sort columns; generate summary reports, charts and pivot tables; add comments; and share the reports with other users.

The Active Technologies API allows you to embed active reports and dashboards in Web applications by programmatically passing data and configuring the visual display of information. The API provides a dynamic way to generate a wide variety of active reports and dashboards without the need for servers or other specialized technologies, thus ensuring a low cost of deployment. The embedded interactive engine in every active report and dashboard allows for a scale-free distribution to an unlimited number of users. There is no need for a backend server because the active reports engine executes all user interactions locally in the browser.

The Active Technologies API comes in a single JavaScript file that can be placed anywhere on the Web server or application server. Application data is passed to the Active Technologies API in the form of an array. You can create an array using any language such as PHP, Java, C, Python or Perl.

The array should be created in the data stream that is used to create dynamic (DHTML) pages. For example, if the data resides in MySQL database, a developer can create a PHP page on the Web server that contains a script to query the database, pass the data, and create the JavaScript array. The PHP page will also create the final HTML page that contains the data and API calls for the customized active report or active dashboard. You can also build fully parameterized active reports because PHP pages can receive and process parameters.

You can simply include the JavaScript file for the Active Technologies API directly in an HTML page. For example:

```
<html>
<head>
<title>My Report</title>
</head>
<body onload="function_name();" >
  <table>
    <tr>
      <td><DIV id="Report1"></DIV></td>
    </tr>
  </table>
<script type="text/javascript" language="JavaScript"
src="location_of_file_on_server/ActiveReportAPI.js">
...
</script>
</body>
</html>
```

In order to use the Active Technologies API, create a function and initialize the Active Technologies API by assigning a new `activeReport` object to a variable. For example:

```
var ARobj = new activeReport();
```

To create a report, you can allocate a new `arGridObject`. For example:

```
var NewReport = new arGridObject();
```

The Active Technologies API expects your data to be in the form of a multi-dimensional array. For example, we can use the sample array as below, which has four records and three columns.

```
var mydata = [ ["Germany", "model 1", 10000],
["England", "model 2", 478457],
["England", "model 3", 23905],
["France", "model 4", 89598] ];
```

In order to add the columns to the new report, you can use the addColumn method. For example:

```
var col1 = NewReport.addColumn("COUNTRY", "Country", IBISTR, 0);
var col2 = NewReport.addColumn("MODEL", "Model Cars", IBISTR, 1);
var col3 = NewReport.addColumn("SALES", "Total Sales", IBINUM, 2);
```

Then bind the data array to the report:

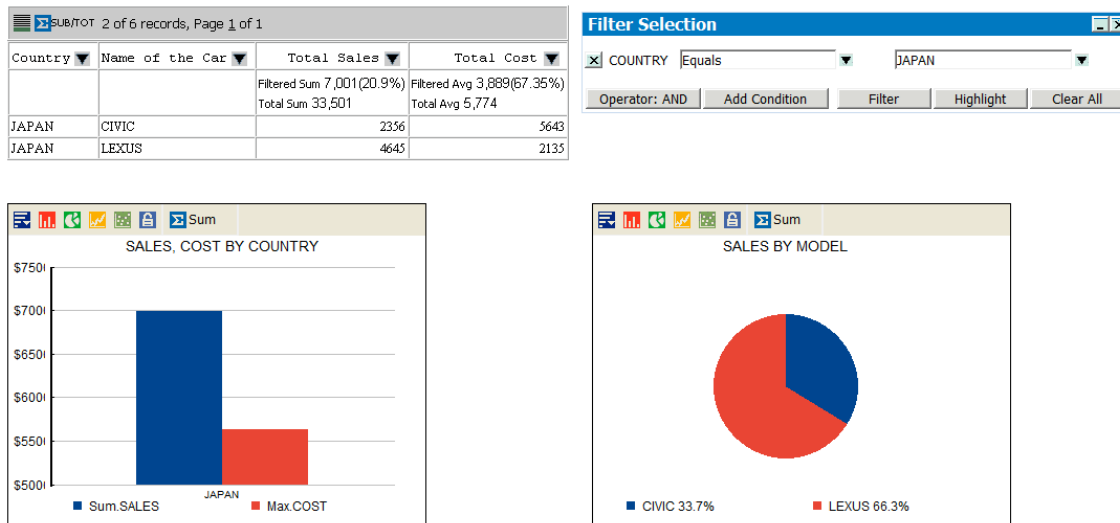
```
NewReport.addDataArray(mydata);
```

Next, tell the API what HTML element to put the report into. In the following example, we can simply add to the <div id='Report1'>:

```
AObj.addToReport(NewReport, "Report1");
```

With very minimal coding, we have quickly created an active report with the interactive menu already embedded.

With a few additional steps, you can change the display of the active report to any type of chart or pivot table to turn the report into the active dashboard as displayed in **Screen 1**.



**Screen 1:** Active Dashboard created using Active Technologies API