Getting Started With WebFOCUS Visual Discovery Analyst Edition (AE)
by Adam Lotrowski

WebFOCUS Visual Discovery Analyst Edition (AE) is a new solution that extends Information Builders’ WebFOCUS platform with advanced visual analytics capabilities. The tool features analytics and dashboards composed of various interactive and interconnected charts that show trends, relationships, and other measured values.

A fundamental concept to WebFOCUS Visual Discovery AE is the in-memory data model, which enables high-performance insights and guided analysis. Behind the scenes, Visual Discovery’s data pool is initially loaded from any combination of data sources and then highly condensed into a proprietary, encrypted format for use by business analysts and data scientists.

The three steps that make up the Visual Discovery AE analysis process are data preparation, guided visualization, and information sharing. Leveraging WebFOCUS, any data source can be accessed with InfoAssist or Report Painter to prepare data using a new output format, VISDISAE.

The resulting datasets are then imported into the Visual Discovery AE development environment where multiple dashboards with various charts can be created and used. Once complete, projects can be published to Microsoft Internet Information Services (IIS) Web servers. This provides interactive access through a Silverlight viewer using popular browsers such as Internet Explorer, Firefox, Chrome, and Apple Safari.

Accessing and Preparing the Data
Once the appropriate Master File Descriptions have been created for any given data source, create a WebFOCUS procedure to review all the available data (be sure to set the record limit returned). The best data for visual charting and analysis typically are fields that identify independent entities. These entities are evaluated by related fields whose values can be grouped together into various dimensions or attributes, as well as fields that contain different metrics or measures.

For example, using the WebFOCUS Retail sample, there is a record for each individual sale ID, which can be grouped by other fields representing customer demographics and product categories that are measured by quantity sold, discounts, revenue and MSRP (Screen 1). Once prepared, set the output to the VISDISAE format, run the procedure, and save the data to a directory that is accessible for later use.
In previous versions of WebFOCUS, the format VISDIS can be used, but be sure to open the output text file and remove the second row, which contains data type information (S for string, I for integer, etc).

Screen 1

Visualizing and Analyzing the Data
Visual Discovery AE includes an intuitive workbench for preparing, visualizing, and analyzing data. Open the Visual Discovery Workbench AE and load the dataset generated by the WebFOCUS FORMAT VISDISAE procedure.

Although referenced here as a single source of data, one or more datasets can be loaded and linked together in the Workbench environment. Once all data is loaded, and placed in memory, users can begin to design multiple pages of associated charts and graphs that are automatically linked together by common attributes and easily identified by default color-coding. Various options are built in to select and filter data – all while seeing how the information intersects. (Screen 2).
Sharing Work with Other Professionals

All Visual Discovery AE projects can also be deployed to a wider audience. As part of the publishing process, there are a few options in how to package and deploy projects. One of the most secure and efficient ways to do this is to embed the data. Rather than also deploy the source data, the data itself can be stored in encrypted form within the project, similar to Active Technology, which allows for faster processing and load times.

Nonetheless, end users are then presented with the same built-in tools that designers have in the Workbench AE environment to select and filter data, as well as options to change presentation and export snapshots to PDF and Microsoft Word (Screen 3).
Visual Discovery AE provides a means to dig into the data, gain insights and share them with other interested parties. The tool accelerates access to information, allowing users to slice and dice multidimensional data for visual analysis with interconnected charts and graphs. Charts and graphs can be as simple as a pie chart, as complex as a parabox (Parallel Box Plots), or as cutting-edge as a heat map with complete inner charting drill-down and back capabilities.

WebFOCUS 8: Data Integrity and Performance Management Enhancements

The newest release of WebFOCUS addresses what we at Information Builders have identified as the four critical elements of smarter business decisions – business intelligence, performance management, advanced analytics and data integrity. These are the four elements that organizations move through as they collect, process and act on business data.

In the January-February 2013 edition of WebFOCUS Newsletter, we covered how WebFOCUS 8 helps organizations leverage business intelligence and advanced analytics to draw insights from data to make strategic decisions. This time, we will focus on the other two elements of smarter decisions – data integrity and performance management.
The Importance of Data Integrity

The importance of data integrity is self-evident to any company that has ever had to wade through reams of data to correct bad information and eliminate duplication. Bad data typically results from the manual processes involved in entering data into corporate systems. Manual processes are prone to error because people involved in repetitive tasks inevitably make mistakes, no matter how diligent they are.

Bad data has a tendency to proliferate, potentially polluting a company’s databases as the information gets passed around and is duplicated. The problem is exacerbated when the data is shared between partners, suppliers and customers in the business-to-business transactions common to today’s business environment.

The most common data integrity issues involve incorrect customer addresses, multiple entries for the same customer, and missing or incorrect e-mail addresses and phone numbers. Failure to correct this data leads to replicating inaccuracies in various reports and documents. The more they are replicated, the harder they are to correct.

Cleansing data is expensive and may involve hiring a third party to analyze and filter it for accuracy. But even worse is letting the data proliferate, which can lead to added costs as a result of duplicate mailings and mailings sent to outdated addresses, as well as inaccurate reports and invoicing, and delays in product launches. In the medical field, bad data can even become a life-and-death situation.

Data Profiling Integration

WebFOCUS 8 is the first business intelligence platform to incorporate data profiling, real-time data quality control, and master data management (MDM) components. Through integration with the iWay Data Profiler, WebFOCUS 8 delivers functionality to gather statistics about enterprise data for “data profiling.” Also referred to as “data recovery,” data profiling helps sift through their data to identify, prioritize, and correct any issues or errors that turn up.

Data profiling addresses questions such as the following:

- What are the data’s primary characteristics?
- Who created the data and how?
- Who accesses it most frequently?
- What is the data used for?
- How clean is the data?

WebFOCUS also integrates with the iWay Master Data Center for master data management. MDM uses a combination of processes and technologies to create a single record, known as “golden record,” for use as a primary frame of reference within the enterprise. The golden record contains recognized values from multiple sources of the same information to store complete, consistent and accurate data in a centralized hub.

This process ensures the data stored centrally is accurate and valid, thereby preventing the spread of bad data and preventing the problems that bad data causes. By integrating a comprehensive
data quality solution into WebFOCUS 8, Information Builders is helping enterprise customers extract as much value as possible out of their data.

**Performance Management Enhancements**

Data integrity works hand in hand with performance management to link strategic financial goals with the execution of business strategy. The ultimate goal is to improve an organization's overall performance and – ultimately – the bottom line. To help companies accomplish this, WebFOCUS 8 features a new integrated version of Information Builders’ Performance Management Framework (PMF).

PMF employs a host of metrics and tools to set performance and accountability objectives and ensure they are managed properly. By leveraging PMF, organizations can take steps to correct inefficient or unproductive processes and operations before they become a problem.

For instance, a sales manager can create a custom dashboard with maps and graphics to keep track of sales numbers, drill down into detailed information about what leads are being followed and which are being neglected. The manager can even set a cross-departmental view to check on the status of shipments to make sure the logistics side isn’t holding up sales. The ability to study information with that level of detail can make the difference in today’s highly competitive market.

The PMF platform includes cutting-edge capabilities such as metric blogging, print-quality publishing, mobile alerts, deep analytics, setup wizards, and customizable end-user dashboards. Users can leverage PMF to manage and communicate corporate strategies to employees and partners directly from a Web browser. PMF enables users at any level to create their own dashboards and mashups to collect and organize the information they need to complete their tasks and make good business decisions.

In WebFOCUS 8, PMF is fully integrated with the BI platform, with performance management functionality linked seamlessly to analytical and operational reporting features. Because it is built on the BI platform, PMF can be adapted and synchronized with other operational systems to accommodate an organization’s changing needs as they occur.

PMF is highly customizable, letting users create their own charts, reports and other documents in a number of ways:

- Ability to add existing enterprise reports, charts, and Adobe Flex charts
- Drag-and-drop assembly
- Support for third-party Web 2.0 gadgets
- Broadcasting and synchronization of preferences to other dashboard objects
- Smart caching of dashboard data
- Export/import of gadgets, such as Wikis and RSS feeds

**Conclusion**

Combined with the other critical elements of decision making – business intelligence and advanced analytics – WebFOCUS 8’s performance management and data integrity capabilities
empower organizations to use reliable information to make decisions with a higher probability of success.

Chaining With the HTML Composer
by Barry Solomon

Form controls in the HTML Composer enable you to prompt users for a parameter value. Chaining is a function that builds a relationship between controls to allow associated control values to be populated based on the value selected and populated to a selected control. For example, if the sample CAR file is used, and the value of ENGLAND (for the COUNTRY control field) is selected, the CAR and MODEL controls would be populated to reflect only CARs and MODELS made in ENGLAND.

In this article, I will explain how to implement chaining by providing an example procedure (written in WebFOCUS) that has three parameters (CAR, COUNTRY and MODEL) in the WHERE clause that will be inserted into the HTML Composer page. The report uses the sample CAR file, which you can find in the ibisamp application folder. In this example, chaining will be used to build a relationship between the CAR, COUNTRY and MODEL controls.

Before chaining controls, the controls must be populated with values. This can be accomplished in the following ways:

- Using an embedded procedure
- Using an embedded procedure with modifications
- Using an external procedure
- Using static values

Using an Embedded Procedure
In order to set control properties (populate values in a control), click on the "Parameter" tab at the bottom of the HTML Composer page. See Screen 1.

Screen 1

In the Parameter window, right-click on the control. A popup menu will appear. From this menu, select the "Properties and settings" option. From the "Properties and settings" window, click the "Embedded procedure" radio button.
When embedding a procedure, the following code will be inserted into the "Properties and settings" screen (text area control) after choosing a master file such as the CAR file. To select a master file, just click on the first ellipsis button (...). **Screen 2** shows the "Properties and settings" window.

**Screen 2**

```plaintext
TABLE FILE CAR
SUM FST.CAR.ORIGIN.COUNTRY
   BY CAR.ORIGIN.COUNTRY
// TODO: Add your filters here to replace defaults
ON TABLE PCHOLD FORMAT XML
END
```

**Using an Embedded Procedure with Modifications**

The procedures for using an embedded procedure with modifications are the same as using an embedded procedure (described above) with the exception that the procedure to retrieve the data that fills the control will be modified by the developer. An example of a modified procedure would be as follows:
TABLE FILE CAR
SUM FST.CAR
BY CAR
BY COUNTRY
ON TABLE PCHOLD FORMAT XML
END

**Using an External Procedure**
When using an external procedure, the developer can leverage the full core FOCUS language. However, the final table request, which will provide the values and text to the control, should be modeled similarly to the procedure above. It should be noted that NOPRINT statements should be avoided in the final table request. The final request is used only to fill the control. NOPRINTs will cause issues with how the HTML Composer reads the file (data in XML format) that is created.

**Using Static Values**
Static values are just a list of values and text that can be added using the "Properties and settings" pop-up window for the control.

**The Chaining Process**
Chaining is a very simple process that can be accomplished by simply clicking the checkbox for each column under the "Chain Control" column in the "New Parameters" window when inserting a report/graph or creating a hyperlink. Screen 3 will automatically pop up when the report is inserted on the canvas, or when a hyperlink is created on the canvas.

Please note that when creating multiple chains of controls, there is an option in the "New Parameters" window, called "Chain Separator," which can be positioned by selecting the row and using the up and down arrows to move the Chain Separator between the various distinct chained controls.
If you select the "Properties" tab, you will see the following, which indicates chaining:

When you right-click the arrowhead on the line between listbox1 and listbox2, the line will turn blue and a pop-up menu with the options "Break Bindings" and "Properties and settings" will appear. By clicking on the "Break Bindings" option, the developer can remove chaining between the two controls.

When choosing the "Properties and settings" option, Screen 4 appears.
The key to chaining is that the selected value(s) in `listbox1` will be compared to the "Resolves parameter" in the Embedded or External procedure (displayed on Screen 4) using the "Parameters compare operator," which is defaulted to "equal." The "Resolves parameter" truly is the key to making chaining work.

When embedding a procedure, the "Resolves parameter" must be the name of a column that is listed in the master file.

*If the embedded procedures have been modified,* you no longer can reference a column in the master file that is not already specifically notated in the embedded procedure. The column or parameter used to set the "Resolves parameter" must be notated in the modified embedded procedure. In an external procedure, the "Resolves parameter" must also refer to a parameter or a column that is notated in the external procedure.
The example below is of an embedded procedure/external procedure for the 2nd control (CAR):

```
TABLE FILE car
SUM FST.CAR
**BY COUNTRY**
BY CAR
ON TABLE PCHOLD FORMAT XML
END
```

```
OR
TABLE FILE car
SUM FST.CAR
**BY COUNTRY**
WHERE COUNTRY EQ &CNTY
ON TABLE PCHOLD FORMAT XML
END
```

In the examples above, the bolded statements are examples of notated columns or parameters that have been added so that the "Resolves parameter" can be set correctly in the "Properties and settings" window.

Please note that when setting the "Resolves parameter" to a parameter, just type in the parameter name (i.e., type in "CNTY") in the Resolves parameter text box.

It must be noted that when modifying an "External procedure", the developer can make changes/modifications in the Text Area Control under the Display field (Screen 5) in the "Parameter and settings" pop-up window.

The changes/modifications can be saved by right-clicking the Text Area Control and then selecting "Save external procedure." Additionally, the external procedure can be modified outside of the HTML Composer. When the developer goes back to the HTM page, just right-click on the Text Area of the "Properties and settings" window and select "Reload external procedure" to reopen the procedure and use the modified procedure.
There Are Dates, and Then There Are Dates
by Art Greenhaus

Anyone who has ever looked at the various format options for fields has noticed multiple date-oriented formats. And surely you’ve noticed there are also numerous subroutines dealing with extraction and conversion of date fields. Have you ever wondered why there are so many format and routines? Well, here’s why.

Legacy Dates
Way back, when computers were first introduced (big machines, not the PCs we’ve come to associate with computers), data was stored in one of two ways. It could either be stored as a number (with some internal representation), or as a character string. No specific date format existed.

As a result, something meant to represent a date was either stored as a number (such as 20130101 [20 million, 130 thousand, one hundred and one]), or as a string of characters (20130101 [the characters 2, 0 [zero], 1, 3, 0 [zero], 1, 0 [zero], 1]). In either case, there was nothing intrinsic in the value indicating it was a date. It was up to the program that used this field to understand that it was intended to represent a date.

Thus, if it was a number, you could subtract from or add to it (20130101 less 1 became 20130100), or as a character string, it could contain non-numeric characters (20130101 might be store [incorrectly] as 20130101 – using the lower case ‘L’ instead of the character ‘1’). In some cases, an
invalid date specification was deliberate, as, for example, when a company used an internal calendar of 13 months (12 months of 30 days, and one month of 5 or 6 days).

To display these "legacy" date formats, the FOCUS language has a format with EDIT options of Y, M and D. Thus, a format of I6YMD was used for a field stored as a number (here an integer), where the first two characters represented the low order year digits (13 for year 2013), the next two characters represented the month, and the last two characters represented the day.

To indicate a four-digit year, 'YY' was used. But this was just an EDIT option, to insert the separator character between components. It did not affect the value. This meant that a date with an invalid value would be displayed with slashes. Thus, a value of 123456, with a format of I6YMD, would display as 12/34/56.

Some “simple” intelligence was built into the format, making it possible to automatically translate the month component to the correct month name or abbreviation – if the value was valid. If not valid, the value itself was displayed. So, a value of 120101, with a format of YMDT, would display as '12 JAN 01', while a value of 123456 would display as '12 34 56'.

Because the EDIT options did not affect the value, manipulation of the components of these legacy dates was not automatic, but up to the program that used it. Even worse, if the date represented was in anything other than a year-month-day order, sorting on that field was not chronological. If the dates were represented as month-day-year, all January dates were sorted before all February dates, without regard for the year. And, invalid date representations (such as those with a lower case L instead of a character 1) could not be correctly interpreted at all.

To support date manipulation, many user-written subroutines were provided, each of which had the logic built-in to correctly interpret the value provided as a date. For example, the AYMD routine will accept a numeric date, in a YMD or YYMD layout, and an increment, and produce the incremented date as a result. But, since the input is just a number, the following must be remembered:

1. Only a numeric input date is supported
2. The components of this date must be year, then month, then day
3. The input date must be valid (no February 30)
4. The increment must be numeric
5. The result is also numeric, in a year-month-day representation

Other routines are also available to handle either alpha or numeric representations of dates, and to perform various operations, but each has its own restrictions.

Legacy dates, and their routines, are the only dates directly supported by the Dialogue Manager because all fields are either numbers or character strings.

**Smart Dates**
To overcome issues with legacy dates, a new date format was introduced. This format stored the date as an offset from a base date. Initial support for these types of fields was provided with “Smart Date” formats. These formats only indicated the way the value should be displayed; the internal representation was left to the underlying engine. Thus, a field that should display only a two-digit year, a month and a day, might be given a format of YMD. Since the actual data was an offset from the base date, this had several advantages:

1. No invalid dates were generated (any offset is valid).
2. Sorting was based on the offset value, so it was always chronological, regardless of how it was displayed.
3. Incrementing or decrementing could be done by addition or subtraction, as it was done using the offset.
4. Duration (days between two dates) could be done by simple subtraction.
5. The underlying engine handled all logic, so it was not up to the coder, and was consistent.
6. It was possible to change component order by changing the format, but without changing the data.

Since we could easily determine the components of both legacy and smart dates, conversion could be done with a straight assignment or one to the other:

SMARTDATE/YMD = LEGACYDATE;

To further help with this conversion, several SETs were added to the language:

DATEFORMAT: When the individual components could not be determined by their value, what was the component order? MDY is the default.

For example, if I assigned a value of '010203' to a smart date field, which was the month, which was the day and which was the year? Using the default, this represents January second.

DEFCENTURY and YRTHRESHOLD: When converting a two-digit legacy year to a smart date, what century digits should be used? The default is 19 and 00.

Since smart dates are stored as an offset, the century is built in, based on the offset. By using the DEFCENTURY and YRTHRESHOLD, the first year of a 100-year span is indicated, and the two-digit year is taken as falling in that range. Once an offset has been stored, DEFCENT and YRTHRESH have no effect, as the offset indicated the century. This can be set at command level or in a MFD at field or file level.

DATEDISPLAY – Should an offset of 0 display the base date? The default is OFF.

By default, a zero offset will display as blank. If it really represents a date, setting this to ON will display the base date instead.
**Date-Time Dates**

With the widespread acceptance of SQL, the final format in the date/date-time progression is the true date-time format. Much like the smart date format, the date and the time are stored as an offset from a base date-time. This has all the advantages of a smart date field, but can also store a time component.

Fields declared as date-time, are indicated much like a smart date, but with a leading _H_ in the format. This also allows a specification for a time component. Times can be specified down to the microsecond level (if stored in the value).

With date-time values, you can display time only, which allows multiple options; date only; or both date and time. If specifying a time-only format, the components must begin with either hour, minute or second, and can continue, in order, as far as desired: hour, minute, second, millisecond, microsecond. Thus, if a date-time field has the value January 2, 2003 2:05:27.123456, you can display the time in numerous ways:

- **HHIS** 02:05:27 (hour with leading zero, minutes with leading zero, seconds with leading zero)

**Note:** Leading zero suppression is only available for the first component given.

- **HHISsA** 02:05:27.123AM (adding milliseconds and AM/PM indicator – AM/PM requires full time specification)

- **HiS** 5:27 (minutes dropping leading zero plus seconds)

- **HHISsma** 02:05:27.123000am (hours through micro-seconds, with AM/PM indicator)

If you display the date plus time, the time component is limited to two characters: one to represent how much time detail to display, and one to indicate whether an AM/PM indicator should be supplied.

- **HYYMD** 2003/01/02

- **HDMYYsA** 02/01/2003 2:05:27.123AM (date and time, through milliseconds, with AM/PM indicator)

One thing to bear in mind when setting a date-time field is that a date-time field has two components. Which represents the date and which represents the time may not be obvious. As a result, the DT function should be used, when giving a literal date-time value in a DEFINE/COMPUTE or WHERE condition. To SET a field to a date-time constant, use the DT function, or use the HINPUT function.

Both the following do the same thing:
However, if the date being set were stored in an alpha string, only the first routine would work because the DT function expects a constant value:

\[
\text{DTCONST/A30} = 'January 2 2003 2:05:27.123456';
\]

\[
\text{DT/HYYMD} = \text{HINPUT}(30, \text{DTCONST}, 10, 'HYYMD');
\]

As a final caveat, when only using the date portion of a date-time field, the time is taken as zero. If the real date-time values contain non-zero times, this may cause an issue when compared against a date range. For example, if a field had the value 'January 2 2003 2:05:27.123456', a test for dates from January 1 to January 2 would not retrieve this record because the January 2 end-point would be taken as time 0.

To correctly include all January 2 times, it should be coded as \( \text{GE January 1 and LT January 3} \). Obviously, this is not the true way to request the date range, as the year specification is omitted, but is only for demonstration purposes. To truly test for the date range, the code should look like this:

\[
\text{WHERE date GE DT(January 1 2003) AND date LT DT(January 3 2003)}
\]

Note the use of the \( \text{DT} \) function to specify a date-time value, and the use of \( \text{LT} \) in the second test. \( \text{FROM/TO} \) would not work, as the range given is inclusive of both start and end dates.

**Integrating Salesforce With WebFOCUS Using the New Query Adapter**

by Clif Kranish

WebFOCUS 8 provides integration with the popular Salesforce.com cloud-based customer relationship management (CRM) solution, allowing users to easily retrieve and analyze information from Salesforce applications.

One way Information Builders makes this happen is through our new Query Adapter for Salesforce.com, which lets users source data from Salesforce applications for Business Intelligence purposes.

The new adapter complements the iWAF (iWay Adapter Framework) adapter, which provides bi-directional integration with internal and external systems. For a refresher on the iWAF adapter, refer to the article “Integrating Salesforce.com With WebFOCUS” [http://www.informationbuilders.com/new/newsletter/13-01/3clif](http://www.informationbuilders.com/new/newsletter/13-01/3clif) in the January-February 2012 issue of the *WebFOCUS Newsletter*. 
To use the Salesforce.com Query Adapter, you need a DataMigrator Server, Release 7.7.04 or later, or WebFOCUS 8.0 and a license for the adapter.

Before you can use the adapter, you need to install it. So here is what you do: Go to the Data Management Console or Web Console using an administrator user ID, connect to your server, and then expand “Adapters, Available, ERP.” Next, right-click on “Salesforce.com” and click “Configure.”

Enter your Salesforce.com user ID (email address), your password and security token as a single entry. Then click “Configure.”

Next, you’ll need to click “Create Synonym” to create a synonym for a Salesforce business object. This should cause the Select Synonym Candidates for Salesforce.com window to open. Here you see a list of all the Salesforce business objects for your organization, including any custom objects that are available. See Screen 1.

Check the checkbox in front of “Account” and click “Create Synonym.” The Create Synonym for Salesforce.com status window should open and display the status “Created Successfully.” Next, click “Edit Synonym,” after which you should see all the fields in the Account Object for your account.

Custom fields appear at the end of the list, and they are the ones with names that end with “__C.” See Screen 2.
To see your Salesforce.com data, right-click under the name and select “Sample Data,” which should cause a report to appear that looks something like Screen 3.
Once you have created the synonym, you can use a Salesforce business object as a source for a DataMigrator flow or a WebFOCUS report.

All aggregation, sorting and filtering operations can be performed on the retrieved answer set. See Screen 4.

Screen 4

As you can see, setting up the Query Adapter for Salesforce.com is a straightforward process. Soon, you’ll be retrieving and analyzing information from your Salesforce.com application as if you've been doing it forever.