Paper 061-2013

SAS® BI Dashboard: Interactive, Data-Driven Dashboard Applications Made Easy

Scott Sams, SAS Institute Inc, Cary, NC

ABSTRACT

The latest release of SAS® BI Dashboard, 4.31 M2, gives you powerful new functionality for designing "Dashboard Applications." Dashboard Applications are a set of interactive dashboards that present a data-driven story to the user. With 4.31 M2, dashboards can link to other dashboards and pass the current click context as parameters, guiding your users through their data with customized dashboard presentations. You can also integrate custom Stored Process, and have its generated data presented by a custom dashboard. This paper will show you several techniques for building Dashboard Applications, using key enhancements in SAS® BI Dashboard 4.31 M2.

INTRODUCTION

Using SAS® BI Dashboard to build a Dashboard Application will save you time vs. building the application from scratch. Using the Dashboard Builder, you can assemble your application and interactions without having to write any code. If you decide to use custom code, you can expand the power of your application by writing custom Stored Processes to provide data and process user input.

Saving time is crucial. One group built a reporting application using custom Flash programming and it took six weeks to build. The same application was built using SAS® BI Dashboard and it took only a few days.

If you have existing SAS® code, you can leverage it with the BI Dashboard Stored Process data provider, allowing you to put a "pretty face" on your data with graphs and tables.

EXAMPLE DASHBOARD APPLICATIONS

To give you an idea what is possible with BI Dashboard, we will present examples from several Dashboard Applications solving real business problems.

Figure 1. Customer Smart Meter application. This application uses the following techniques discussed in this paper: Navigation bar, Status messages, Embedded help, Time frame control, Custom filtering.

Figure 2. Market Optimization. This application uses high-level "data entry" to save user input for later use in the application.

Figure 3. Revenue adjustment maximization. This application uses custom filtering/paging.

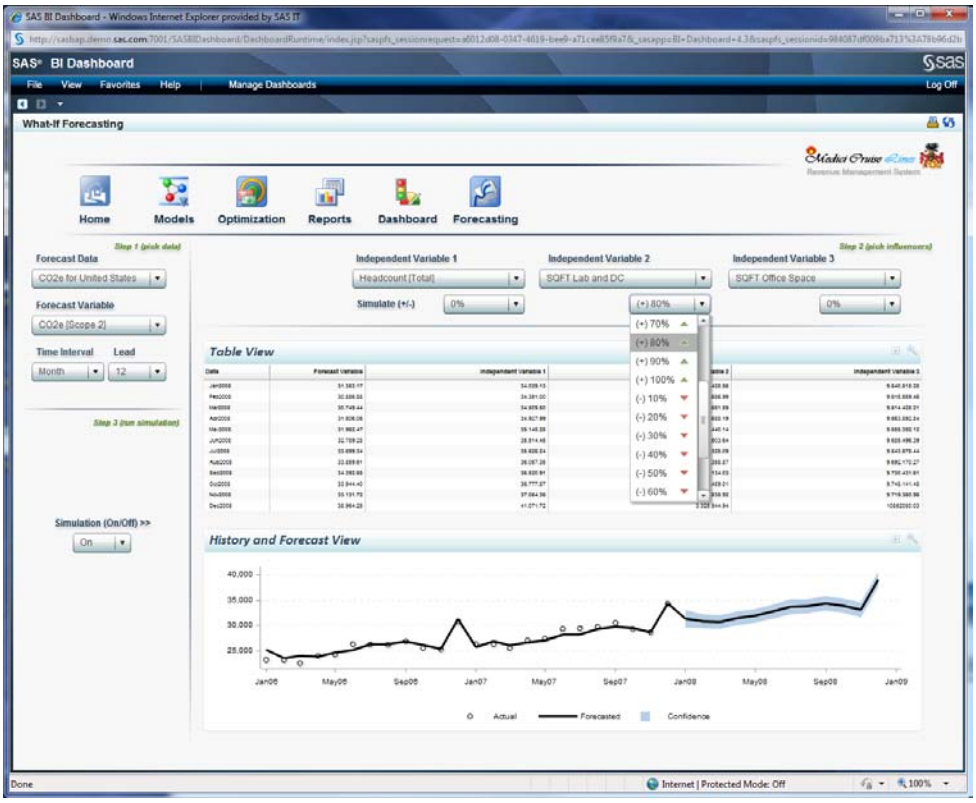

Figure 4. Forecasting and "what-If" analysis. This application uses dynamic prompts with a gauge as an input control to run forecasts .

Figure 5. Financial reporting dashboard. This dashboard uses multiple drill downs and time frame control.

GENERAL DASHBOARD APPLICATION TECHNIQUES

POWERING DASHBOARD APPLICATIONS WITH THE STORED PROCESS DATA PROVIDER

The Stored Process Data Provider gives you powerful control over processing user input and organizing data to display. You can collect user input and run advanced analytic forecasts and display the results in a graph directly in the dashboard.

The idea is to create a Stored Process that takes one or more parameters. These parameters are used to generate a single dataset that will be returned to BI Dashboard. You can then display this dataset using any of the BI Dashboard visualizations.

DASHBOARD PARAMETER PASSING

For some applications, you may want one dashboard to select or filter through options. Then you may want to pass the current set of parameters to another dashboard for further exploration. This is accomplished through the Linking Dialog. In 4.31M2, you can link an Indicator to another Dashboard and pass parameters that the Indicator knows about.

HIGH-LEVEL "DATA ENTRY" AND PROCESSING

Any time a user clicks on an indicator, your application can pass information to a Stored Process. This information can be stored in a dataset for later use in the application. For example, you can present a table of "leads" that is generated by a forecast. When the user clicks on a particular lead, you can save that lead in a table of "interested leads" for that user, which can be displayed on another dashboard for them to explore at any time. You can also have a stored process to delete the lead when the user clicks the row on that table.

DELAYED INTERACTIONS

Sometimes, you can have a large number of parameters that would be passed to an indicator for display. Normally, every time a user makes a selection in a prompt, that parameter is sent to the indicator, and the indicator is refreshed. This might be undesirable. To prevent this, create two versions of your dashboard: one with everything and one with just the prompts. On this "prompt" dashboard, all of these prompts will go to a single "Custom Graph" indicator. This hidden indicator is powered by a stored process that has one job: to record all the parameters it receives into a dataset, then return it. When the user clicks on this image, it links to your "main" dashboard and passes all the parameters it has been keeping track of. The main dashboard will then load and all the prompts will be populated, as well as the indicator of interest.

BUILDING DRILL DOWN EXPLORATIONS

Drill downs in BI Dashboard are very flexible. You can drill down by using the Linking Dialog to pass parameters to another Indicator, even one based on completely different data. You need only decide on which parameters to use in the drill sequence.

USING CASCADING PROMPTS

Cascading prompts are a powerful and common technique. It is possible to design these using interactions with BI Dashboard. The key consideration with cascading prompts is that the cascade sequence could be "ragged" and therefore all prompts "below" the current one need to be refreshed when a selection is made. Each prompt needs to send a copy of the interaction to each of its "children" in the hierarchy (which includes the indicators displaying the data). The last prompt in the hierarchy only sends one interaction to the target indicators. The picture below illustrates this better. While it might look daunting at first, it will make sense when you understand what is happening. This hierarchy is: Country->State->Year->Quarter. Note: for the third level of the hierarchy (Year) we are using a graph instead of a Dynamic Prompt. A "prompt" doesn't have to be a menu!

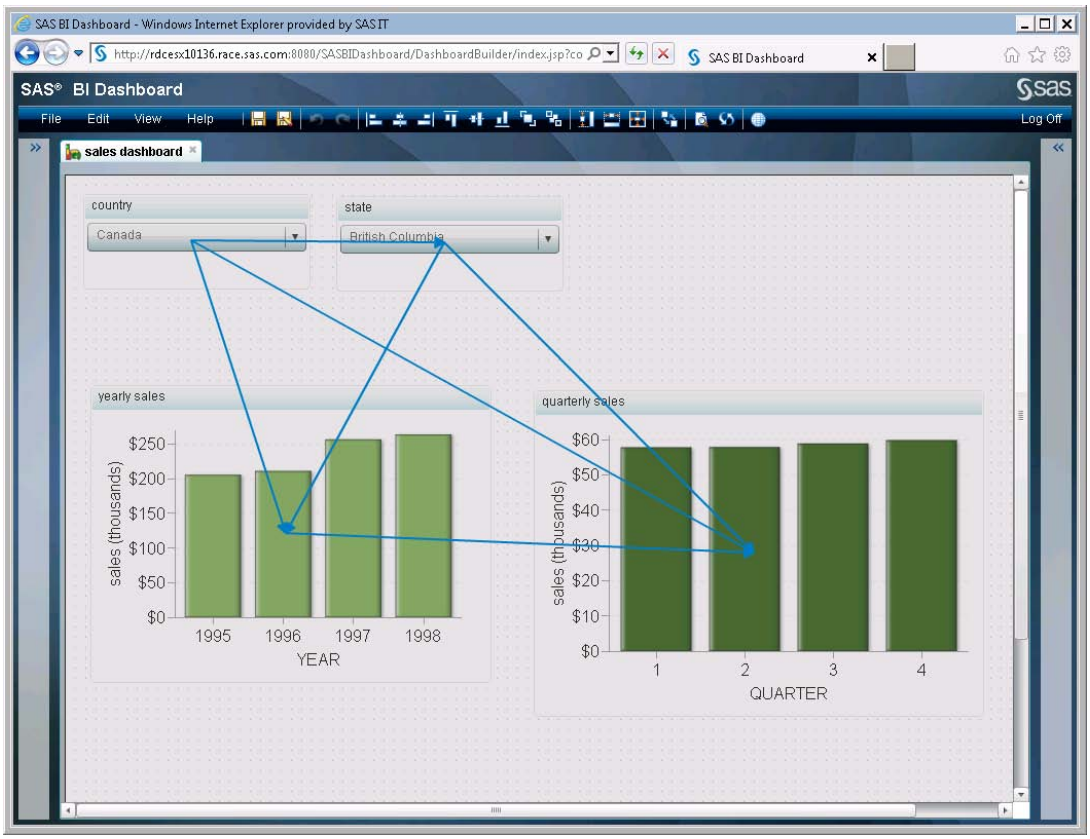

Figure 6. Interactions view of the cascading prompt example

BUILDING COMPARISON DASHBOARDS

A useful new feature in 4.31M2 is the ability to apply default parameters to an indicator. With this feature, it is easy to set up comparison dashboards to compare indicators using the same data model, but with different parameters. For example, if you have a Stored Process that takes the parameter Year, you can create two identical indicators that pass different default parameters to the same Indicator Data. This way you could compare 2012 sales with 2011 sales side by side on a dashboard. This can also greatly simplify dashboard design, as different indicators can share the same Indicator Data. Building comparison dashboards is simple: once you create your "master" Indicator, you just change the parameters and do a "Save As…"

SPECIFIC TECHNIQUES

MINIMAL DECORATION MODE

In the example screenshots in section 2, you will notice that the dashboard application runs in its own window, without extra decorations, looking like a real "standalone" application. To do this, create an .html file with JavaScript to launch a new browser window without decorations, and load a special URL for BI Dashboard that hides its decorations (like when used in the Portal), and load whatever dashboard you like. You need to know the SBIP URL, which you can copy from the browser's URL bar after you click 'Manage Dashboards' on the dashboard you are viewing. The JavaScript to launch this looks like:

```
<body
onLoad="window.open('http://<SERVER>:8080/SASBIDashboard/Director
?_directive=PortletDisplayDashboard&dashboard=<SBIP_URL>&frameWid
th=800&frameHeight=600', 'BI Dashboard',
'width=800,height=600,toolbar=no,location=no,menubar=0,status=no,
directories=no,menubar=no,scrollbars=no,resizable=no')">
```

APPLICATION NAVIGATION BAR

A navigation bar makes the collection of dashboards look like a real application. If you are using minimal navigation mode, it is an essential technique to change dashboards. Each "button" is a separate image with a link to a different dashboard. To make things easy, it is best to build the navigation bar first and to save this in a "template" dashboard so it will be easy to create the other dashboards based off it. Once you have created your other dashboards, it will be more work to make changes to the navigation bar, since you would have to edit all the dashboards. On each dashboard, you can mark the "current" navigation point somehow, either with another image, or change out the image for the current "button." You could also disable the link for the current "button" image.

STATUS MESSAGES

You can use the Dynamic Text indicator to create simple status messages. Your message has the opportunity to change whenever a user loads that dashboard. You just need to create the data behind it and generate your message.

Another use for a status message is within an interaction. The Dynamic Text indicator can be the recipient of any interaction and the message can change any time one of those interactions occurs.

A third usage is to display a title, e.g. "2010 Sales" above a graph. The year would change whenever you change the "year" prompt.

"ALERTS" INDICATOR

You can make your own "Alerts" indicator with the SparkTable indicator. The data behind the indicator can be custom generated from a Stored Process or come from a table that is updated externally.

You could also make a "news feed" indicator that links to external URLs.

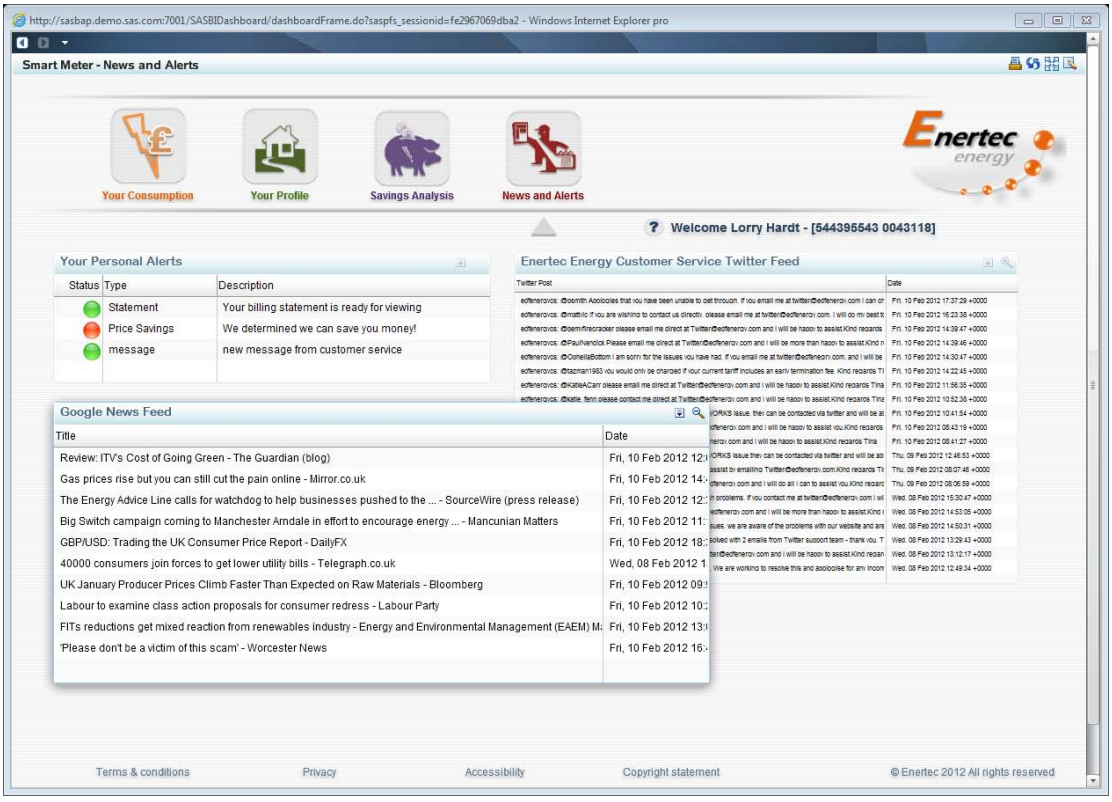

Figure 7. Custom "alerts" indicator built using SparkTable

EMBEDDED HELP

You can provide help to your users several ways. First, you could simply position static instructions using the Text decoration.

Second, you could position a "question mark" image and link it to either an external URL that provides help, or to a

Custom Graph Indicator, which could load an image URL of nicely formatted text and images. With this method, your help will appear in a "pop-up" window inside the application which looks more seamless.

The third method is to use static text with a link to help, just like using an image.

CUSTOM PAGING

You can use a Dynamic Prompt to page results on a SparkTable, e.g. "1-100, 101-200, 201-300, >301" etc. The data for the Dynamic Prompt would need to know the number of rows in the target table, and output the intervals. The Stored Process for the target would take a "page" parameter that would determine which rows to show.

CUSTOM FILTERING

Sometimes, you may want to have a complex filter and not rely on a single parameter. It is possible to create a Dynamic Prompt with a series of custom "where clauses" that could be used to apply a custom filter for the target. Each where clause would be for a commonly used filter, like: "Top 10 Revenue" or "15 Most Profitable." The entire "where clause" gets passed as a single parameter to the Stored Process, which would use the clause to subset the data. The nice thing about this approach is you can save your where clauses in a separate table and add new filters easily.

CONTROLLING A TIME FRAME

When using time data, there are several things you may want to control: the interval size (Day, Week, Month, and Year) and the start and end date. Your Stored Process would aggregate the data differently for each interval size, and limit the results based on the start and end date.

You could use a Dynamic Prompt to change the interval size. There a few options for the start and end date. You could have two Dynamic Prompts to independently set these values. Or you could use a "Chart With Slider Prompt" if you wanted to select a value by dragging a slider on a "big picture" graph. You could also use a "window" parameter, such as the number of weeks/months/years, and just have a start date.

OTHER 4.31_M2 ENHANCEMENTS

EASIER BUILDING OF PROMPTS

Prior to 4.31M2, you had to select a Range when creating a Dynamic Prompt, even if you were not going to use it. Now, the Range is not required so you can build simple prompts quickly. You can add a Range later if you change the type to "Prompt with gauge."

A second feature is the ability to display a label for the prompt. Previously, you would have to create a separate Static Text object and play with the alignment. Now, it is much easier to create great looking forms that line up.

"ALL" OPTION FOR PROMPTS

There is now an option to include an "All" marker at the end of the prompt that will pass all the values in the menu to the target. This works automatically when using InformationMaps, but when using a Stored Process, you need to use the right code to take advantage of it.

The easiest way is to use the %_eg_WhereParam macro, generated using SAS Enterprise Guide®. You should be able to get a copy of this from a Stored Process generated by EG. In this example, yearPrompt will contain either a single year, or multiple years when "All" is selected. The macro automatically generates the where clause portion and can detect single vs. multiple values.

```
WHERE %_eg_WhereParam(t1.year, yearPrompt, IN, TYPE=N)
```

e.g.: this would resolve to:

```
WHERE t1.year in (2008, 2009)
```

MULTI-SELECT INTERACTIONS

You can now use multi-select on graphs and tables for interactions by holding down the control key. This can be used for comparisons to quickly compare, e.g. different regions.

To take advantage of multi-select in Stored Processes, use the same `%_eg_WhereParam` macro. Then your Stored Process can handle single or multi-select with the same code.

BOOKMARK A DASHBOARD WITH THE CURRENT PARAMETERS

The Dashboard Viewer application has a new feature for users to bookmark a dashboard with all the current parameters applied. (Normally, when you load a dashboard in 4.31 the prompts will be reset to whatever value you selected last.) This enables users to create "favorites" for comparisons, e.g. "2008 Sales" and "2009 Sales" or to bookmark commonly viewed selections, e.g. "U.S. Sales" and "Europe Sales."

AUTOMATIC LOOKUP OF PARAMETER NAMES IN THE LINKING DIALOG

Building links to Indicators and Dashboards just got easier in 4.31M2. Previously, you would have to know the exact name of all the parameters to pass. Now, those parameter names are automatically fetched, so it is much quicker to build links and less error prone.

DESIGNING DASHBOARD APPLICATIONS FOR OPTIMAL PERFORMANCE

1. Reduce the number of indicators on a single dashboard.
    a. You can spread your information across several dashboards and use a navigation bar to assist users.
    b. If possible, make one query do as much work as possible. For example, if you want to display a KPI for five regions, don't make five KPI indicators. Make a single KPI and have your query fetch all the regions you need.
2. Return only the data you need in your queries. Instead of doing a "Select *" pick only the columns you need for the graph/table.
3. Use client side filters where appropriate. If the total number of rows in the table is reasonably small (<2000) make your query return all the rows and use a client side filter.
4. Make sure you are using the latest hotfix for fastest performance.
5. Configure data caching on the midtier. This may not be appropriate if your application processes user input and needs to refresh the data.

CONCLUSION

Building Dashboard Applications with SAS® BI Dashboard saves development time and produces great-looking interactive applications with your data. The techniques and examples explored in this paper will give you an idea what is possible for your business problems, and help you get started creating custom Dashboard Applications.

ACKNOWLEDGMENTS

Special acknowledgments and thanks go to John Green's team, especially Keith Renision and Carlos Sebastiani. They have been producing lots of pre-sales solutions for big customers using BI Dashboard and custom Stored Processes. They chose SAS® BI Dashboard because it looks good, has great functionality and flexibility, and it saves them time. What they have done with the product was more than we imagined possible!

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Scott Sams
SAS Institute, Inc.
SAS Campus Drive
Cary, NC 27513
Work Phone: (919) 531-1076
E-mail: Scott.Sams@sas.com