Introduction

These are the general best practices for buidling visualizations with the **Dashboards** designer (also known as SAP BusinessObjects Dashboards 4.0, SAP Crystal Dashboard Design 2011 or Xcelsius 2008).
This is an updated version of the older Xcelsius 2008 General Best Practices whitepaper which included best practices from Ryan Goodman, Richard Reynolds, and Matt Lloyd.
It also includes best practices from Patrice Le Bihan.

Who Should Read This Guide?

Anyone who is creating visualizations with the Dashboards designer should read these best practices.

About Dashboards (Xcelsius)

The Dashboards designer is a visualization tool for creating interactive visual models (dashboards) based on **high level**, **pre-aggregated** data sets.
It uses a point-and-click design time environment that can easily be used by business users. No programming skills are necessary for creating visualizations.

The Role Of Microsoft Excel At Design Time

Microsoft Excel is only used in the Dashboards designer at design time when building visualizations.

We often are asked what benefits we get from using Microsoft Excel and why do we continue to use Microsoft Excel?
Apart from Microsoft Excel having a wide reach among business users and being easy to use, there are three key benefits that Microsoft Excel provides when embedded inside the Dashboards designer:

1. A Data Model.
   You can enter highly aggregated data directly into the spreadsheet and visualize it, so there is no need for a database. You can also pull in highly aggregated data from external sources, including SAP Business Warehouse (BW), SAP BusinessObjects Business Intelligence (BI) universes, XML and simple Web services.
2. A Calculation Engine.
   In addition to Microsoft Excel having a very familiar and flexible formula language, it also provides a calculation engine.
3. An Eventing Model.
   When a cell changes, we get a “data change” event. This is a very simple model, but we leverage it for more than just calculations.

You can either start with a blank spreadsheet inside the Dashboards designer or you can import an existing spreadsheet.
To export the Microsoft Excel spreadsheet out of the Dashboards designer from the **Data** menu, click **Export**.
The spreadsheet uses the .**xls** format.

The Role of Adobe Flash At Preview Or Export Time

All of the supported Microsoft Excel functions (logic), data and formatting are embedded into an Adobe Flash Shockwave (SWF) file at previewor export time.

So when you distribute your visualization as a SWF, all the end users need in order to view the SWF is the Adobe Flash player.
Dashboards 4.0 Feature Pack 3 (FP3) or later require Flash Player 10 or later.
Dashboards 4.0 SP02 and Xcelsius 2008 require Flash Player 9 or later.

Supported Microsoft Excel Functions

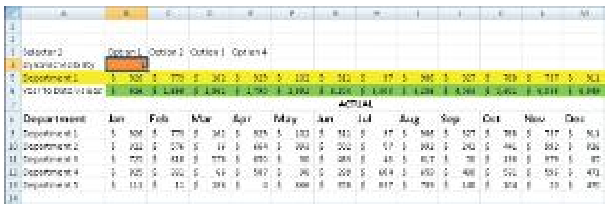
The Dashboards designer supports a lot of the Microsoft Excel functions. Make sure that your Excel spreadsheet uses only the Excel functions that are supported, otherwise when previewing or running the SWF file your visualization may not behave as expected.
Refer to the Supported Excel Functions section in the user guide.

Use Colors, Labels, And Borders To Identify Data Types (Input and Output)

To make it easier for you or other members of your organization to maintain Dashboards models, it is a good idea to use colors, labels, and borders to identify cells or ranges of cells in the spreadsheet and to hint at their use.
It is often a good idea to create a legend in your spreadsheet to say what the different colors represent (input from the dashboard, calculations in the spreadsheet, lookups, and dynamic data) – that way, you can decide on a color scheme that makes sense for your organization or team.

Color	Description
Bright Yellow	Input from the dashboard
Light Green	Microsoft Excel logic (function or calculation) that is read-only in the dashboard
Dynamic Visibility	Dynamic Visibility allows you to show or hide components at runtime

Table 1: Sample Legend



Selector 2	Option 1	Option 2	Option 3	Option 4								
Dynamic visibility	[illegible]											
Department 1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
						ACTUAL						
Department	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Department 1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Department 2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Department 3	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Department 4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Department 5	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Figure 1: Colors and Labels Used to Indicate Spreadsheet Inputs and Outputs

Organize Your Data In A Logical Fashion

To make your spreadsheet easier to understand, it is a good idea to lay out your data in a logical fashion. For example, group related items together – again using colors,
labels, and borders. See Figure 1.
One of the features (introduced in Xcelsius 2008) is the embedded Microsoft Excel spreadsheet at design time. The embedded spreadsheet means you can insert or remove rows or columns, and the components track the ranges they are bound to. It is still a good practice to leave room below or to the right of your data so it can grow over time without having to add/remove rows or columns. See Figure 2.
If you insert or remove rows in an external spreadsheet and import it, the components cannot track range changes.

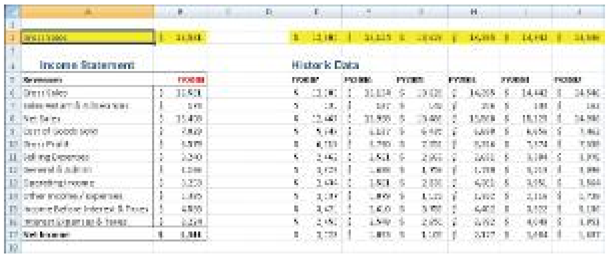


Figure 2: Organized Data with Lookup Row Above the Data so the Data Can Grow Downwards Over Time

Place Frequently Used Data And Logic At The Top Of The Spreadsheet

Often you bind data from your spreadsheet into components, to make it easier to select the data you want and to minimize the amount of scrolling you have to do. It is a good idea to place frequently used or common logic or data at the top-left of the tab in your spreadsheet. See Figure 3.

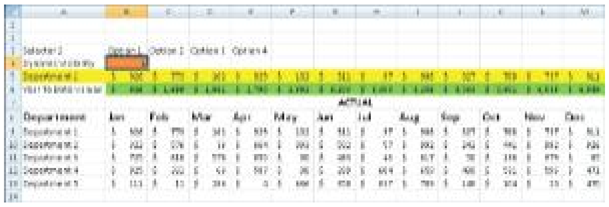
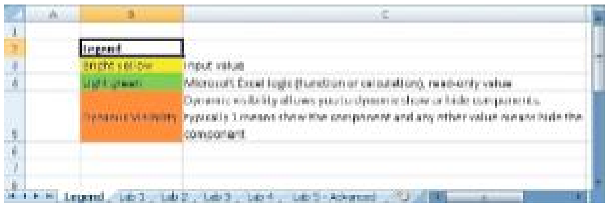


Figure 3: Place Common Data and Logic at the Top of the Spreadsheet

Use Multiple Tabs

If you find that you have to keep scrolling your spreadsheet tab to see your logic or data, you may want to consider using multiple tabs (or worksheets) instead. See Figure 4.



Legend	
Bright yellow	Input value
Light green	Microsoft Excel logic (function or calculation), read-only value
Dynamic visibility	Dynamic visibility allows you to dynamically show or hide components. Typically 1 means show the component and any other value means hide the component

Figure 4: Using Multiple Tabs in the Spreadsheet

Designing And Creating Visualizations

The following are the best practices when designing and creating dashboards.

Design On Paper

Remove yourself from the data you want to visualize. Create a rough drawing on paper away from your computer. This can be useful to get a general idea for the visualization you want to create and also to use to sign off the basic design of your visualization.

Start With An Empty Spreadsheet

Dashboards does not support using spreadsheets that have links to other spreadsheets or have macros in them. So it is a good idea to start with the empty spreadsheet that is embedded within the designer.
Using the empty spreadsheet will also reduce the risk that you use Microsoft Excel functions or plug-ins that are not suppoorted.
Macros, external spreadsheets and 3rd party Microsoft Excel plug-ins are not supported.

Keep The Data And Logic To a Minimum Or Use Hard-Coded Values

The more data or logic you put into your spreadsheet, the larger your generated SWF will be and the longer it will take to open.
Also, the more logic (and nested functions and cell references) you put in your spreadsheet, the longer it can take when the data changes, so the cell values (and related cells values) have to be recalculated.
If you have any data or logic that is not used within your visualization, get rid of it from your spreadsheet.
If you have functions for values that do not change (so are not dynamic or used by what-if scenarios), then to make sure your visualization is more efficient, convert the data to fixed (hard-coded) values using **Copy** and **Paste Special** (as Values) in the spreadsheet.

Add Components

Using the design you created on paper earlier, mock up your visualization using components. At this point, there is no need to add data to your spreadsheet; you are just creating the basic visual layout for your visualization.
However, it is a good idea to share your design at this point as a second stage of feedback before you start adding more complexity.

Add Some Dummy Data To Test Components And Interactivity

Add in some dummy data to your spreadsheet – just enough so that you can test your components and interactivity. Again, share your design to get feedback before you add more complexity.

Show or Hide Components At Runtime With Dynamic Visibility Logic (Optional)

Often you want to re-use the real estate within your dashboard by hiding and showing components based on user interactions or data changes.

You can control when a component is shown or hidden by using **Dynamic Visibility** in the **Behavior** tab.

When the value in the Dynamic Visibility **Key** cell matches the value in the **Status** cell the component will be shown, otherwise it is hidden.

There is a sample built into the designer, go to **File** > **Samples** > **User Guide Samples** > **Dynamic Visibility**.

Test the interactivity of hiding and showing components, and again, share your design to gather feedback.

Remove Your Dummy Data

Now it is time to replace your dummy data with actual data or data to be fed from outside of your dashboard.
If the data in your dashboard is of a secure nature, it is a good idea to remove the dummy data from your spreadsheet (but leave the cells and cell coloring as they were). Thus, before the data is retrieved and displayed, there is no data to see until the end user has successfully authenticated.

Export And Test Your Dashboard

Make sure you test your SWF in preview mode and also wherever you will deploy your visualization.

It is important to do this because when running outside of the Dashboards designer, Adobe Flash has certain security restrictions – see the section “Flash Player Security” for more information.

Embed Your Dashboard Into A Web Page (optional)

To embed your dashboard SWF into an HTML Web page, you need to use the OBJECT and EMBED tags. The easiest way to generate the HTML for the Web page is to Export to HT ML from within the designer (which also exports the associated SWF).

There are two optional advanced topics when you export your SWF into a Web page:

1. Pass initial values using Flash Variables:
a. Use the Flash Variables connection in your dashboard to define the Flash Variables names and where to place the values in the spreadsheet.
b. When you export your HTML page the current spreadsheet values are also added into the OBJECT and EMBED tags.
c. For more information about passing in Flash Variables see: http://www.permadi.com/tutorial/flashVars/
2. Pass values in and out at runtime using External Interface:
a. Use the External Interface connection in your dashboard to define the ranges of spreadsheet data that can be read or written to at runtime from outside of the dashboard.
b. If you deploy your dashboard into SAP BusinessObjects BI Workspace these are the ranges that you can use with **Content Linking**.
c. When you use the OBJECT or EMBED tags in a HTML web page you can use JavaScript to listen for changes or to pass in values for the ranges you defined in the External Interface.
d. You can read more about External Interface in the Dashboards User Guide.
e. Here is an example HTML web page that uses External Interface and JavaScript to pass values between two SWF files.

There are certain restrictions to Adobe Flash when you want to call out to JavaScript; see the section “Flash Player Security” for more information.

Flash Player Security

Depending on how you distribute or run your SWF, the Adobe Flash player places certain security restrictions on a SWF that is running. Because the Dashboards designer is an application, we choose not to apply the security restrictions – so you can query data from wherever you need to at design time or preview time.

If you export your SWF to Adobe Acrobat refer to the Adobe documentation as it has its own security settings.

If the Flash Player Security settings have prevented the SWF from accessing external data the dashboard will do its best to identify the issues and suggest a workaround.

If you see an error number refer to the list of Adobe runtime errors at
http://livedocs.adobe.com/flex/2/langref/runtimeErrors.html.

Running A SWF From Your Desktop - How To Make The SWF Trusted

These instructions only apply when you are running the SWF on your desktop.

If you export to a local Microsoft PowerPoint, Microsoft Word, HTML file, or to your desktop to run the SWF, you may find the SWF does not work if you try to retrieve data or try to navigate to a Web page because of the Adobe Flash security restrictions.
To run this SWF on your desktop, you need to make it trusted, so it can access Web sites or local data.

1. Access the Flash Player Settings:
   a. When running a SWF you can **right-click** and from the menu choose **Global Settings...**
   OR
   b. On Microsoft WIndows go to **Control Panel** > **Large Icons** > **Flash Player (32-bit**
2. The **Flash Player Settings Manager** opens
3. Click on the **Advanced** Tab
4. Within **Developer Tools** click **Trusted Location Settings...**
5. Click **Add...**
6. Click **Add Folder...**
7. **Browse** to the folder that has the SWF file you want to run and click **OK**
8. Click **Confirm**
9. Click **Close**
10. Close the Flash Player Settings Manager
11. (Re)-Load the SWF file and it should now be trusted

Running a SWF File That Is On A Web Server - Always Add A Cross Domain Policy File To Your Web Servers

These instructions only apply when you are running the SWF from a web server.

If you run the SWF file from a web server (either a local or remote server) by default the Flash Player Security Settings may prevent your SWF from accessing data outside of your SWF.

To fix this you need to add a cross domain policy file to the root folder of your web server.
It is a best practice to add a cross domain policy file to every web server that you want to access data from including the one that the SWF file runs from

The root folder differs depending on which web server you are using.

The root folder for Apache Tomcat is: ***<TOMCAT_HOME>*****\webapps\ROOT**

When the cross domain file is in the root folder test you can get to it with a web browser:
http://*yourwebserver:port*/crossdomain.xml

Here is the crossdomain.xml file for the SAP BusinessObjects 4.0 FP3 web server.

It allows any SWF on any domain to access data on the web server (so it behaves in a similar way to if someone was running the SWF from the desktop).

Factors That Affect Dashboard Load Time

Lower numbers are better:

1. Queries or connections that are set to **Refresh Before Components Are Loaded**.
   This is the default setting for a query.
   If you have any queries or connections set to refresh before components are loaded the **Initializing** message phase stays up until they have all completed.
2. Excel cells
3. Excel calculations
4. Components
5. Bindings from components to Excel (also size of ranges)
6. Dashboard file size

By default the maximum number or rows you can refer to in a formula or in a binding is **512** rows.
You can adjust this in **File** > **Preferences** > **Excel Options** (max recommended is **2000** rows) but this may affect the performance of your dashboard.

Recommendations

Run Queries And Connections After The Dashboard Loads

The default is that queries run when the Initializing message is shown as the dashboard loads.

The Initializing message stays up until all of the queries or connection that are set to Refresh Before Components Are Loaded have completed.

This can feel slow to the user if more than one query/connection needs to run.

Instead we want the dashboard to open as quickly as possible with no data, then to query for the data and as the data is returned the dashboard updates. This will feel much faster to the user although the overall end-to-end time is similar.

Here is what you should do:

1. Go the Usage options:
   a. For a query navigate to the **Usage Options**
   b. For a connection go to the **Usage** tab
2. Uncheck **Refresh Before Components Are Loaded**
3. Add a refresh button:
   a. For a query add a **Query Refresh Button** (in Universe Connectivity) to the canvas
   b. For a connection add a **Connection Refresh Button** (in Web Connectivity) to the canvas
4. In the refresh button **Behavior** tab check **Refresh After Components Are Loaded**
5. Select the queries/connections to run:
   a. For a query in the **General** tab check each query that is to be run on load
   b. For a connection in the **General** tab check each connection that is to be run on load
6. Optionally hide the refresh button:
   a. Do not use Dynamic Visibility as that may prevent the button from calling the queries/connections
   b. You can place the button behind another control

      OR
   c. Got to the **Appeance** tab and uncheck **Show Button Background**
   d. Go to the **Appearance** > **Text** tab and unheck the **Label**

Apply Microsoft Excel Best Practices

These are the best practices for working with Microsoft Excel in your dashboard.

Avoid:

- array calculations as they can dramatically affect performance:
  1. SUM
  2. COUNT
  3. SUMIF
  4. COUNTIF
  5. INDEX
  6. MATCH
  7. HLOOKUP
  8. VLOOKUP
  9. *etc*
- repeated logic:
  1. =IF(SUM(A2:A100)<1000,1000,SUM(A2:100))
  2. This will run the SUM twice
- any calculations:
  1. They add overhead when loading or interacting with the dashboard
  2. They increase the time it takes to generate the flash file when you export or preview

Do:

- Hard code values wherever possible instead of using formulas
- Define calculations on the server side where possible
- Use server side sorting to order your data and make the results easier to compare
- Use server side ranking to limit the number or rows returned

- Link to a report for detailed drill down
- Use a selector with filtered rows instead of Excel lookups
- Use direct binding from components directly to the results of a query (new in Dashboards 4.0) to bypass the spreadsheet layer wherever possible

Which Charts Best Represent...

Time Series

How a quantitative value evolves over time (Year, Quarter, Month, Date, Hour, Minute).

Think Line Charts or Column Charts.

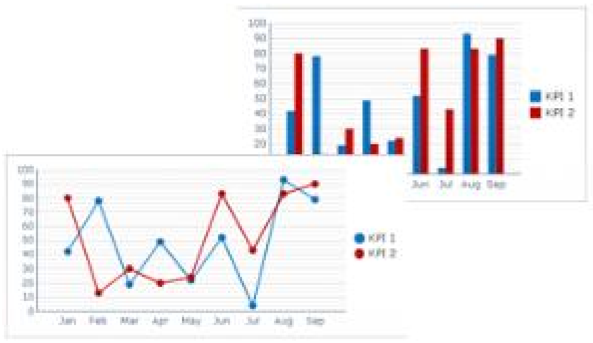


Figure 5: Line or Column charts to best represent Time Series

Contribution Relationship

How much my line of business contributes to the overall revenue of an organization.

Think Pie Charts or Stacked Bar Charts.

Avoid using Pie charts

It is often difficult to compare the relative size of each slice in the pie, it is better to use Bar or Column charts instead.

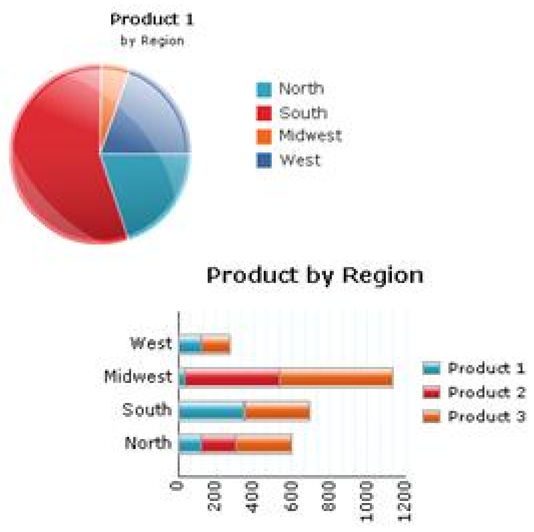


Figure 6: Pie or Stacked Bar charts to best represent Contribution Relationships

Deviation Relationships

Typically actual to plan and/or target.

Think Combination Charts or Line Charts.

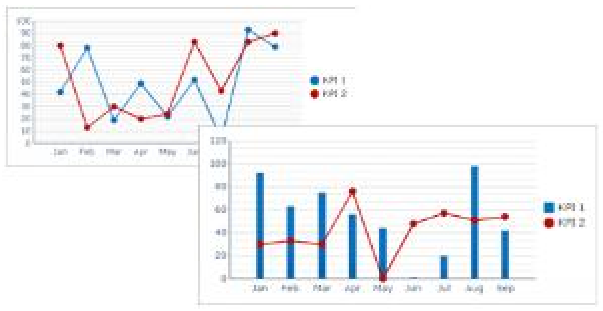


Figure 7: Combination or Line charts to best represent Deviation Relationships

Correlations

How two metrics relate one to another.

Think Bubble Charts or XY charts.

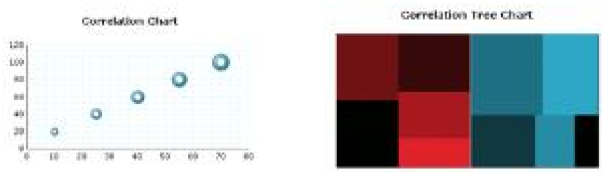


Figure 8: Bubble or XY charts to best represent Correlations

Use Selectors With Filtered Rows Instead Of Microsoft Excel Logic To Copy Rows Of Data

It is better to use server side queries instead of manipulating data in the embedded Excel spreadsheet.

If you do need to copy/filter rows of data in the embedded spreadsheet (which may affect performance) do not use Excel logic as this is very slow because each cell has to be calculated individually.

Instead use a dashboard component:

1. Add a selector to the canvas (List, Combo Box, Labed Based Menu)
2. Bind the labels to the list of labels you want to pick from
3. Change the Data Insertion to **Filtered Rows** and then build the source data (make sure to refer to the same rows as the labels) and destination data
4. This removes duplicate labels from the list of labels in the selector
5. When you select a label all of the source data rows that have that label are inserted into the destination in one go
6. This is more efficient than the equivalent Excel logic to filter with lookups

There is a sample build into the designer, go to **File** > **Samples** > **User Guide Samples** > **Filtered Rows**

Restrict Query Data Set On The Server Side

Use server side logic to filter, sort and restrict your data sets.

You can define sorting and prompts (to filter data) in the Dashboards 4.0 query panel.

You should also consider using server side ranking to limit the data to Top/Bottom rows as well, see Figure 9.

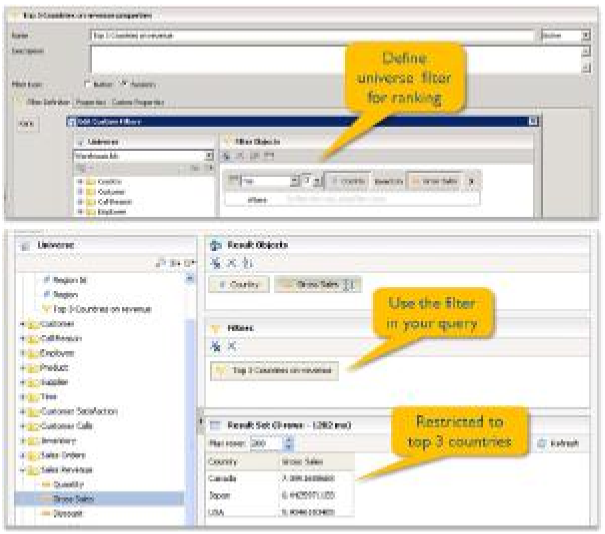

Define universe filter for ranking
Use the filter in your query
Restricted to top 3 countries

Figure 9: Restrict Query Data Set On The Server Side

Link To A Report For Detailed Drill Down

Keep the dashboard high level, two or three levels max.

For additional detailed drill down link to a report:

1. In BI Launchpad right click a report and select > Document Link
2. Add a URL Button to the canvas in your dashboard, the URL is the Document Link
3. Optionally pass report prompts (context) to via the Document Link, refer to the OpenDocument documentation.