Crystal Xcelsius 4.5 Tutorials

Applies to:

Crystal Xcelsius 4.5

For more information, visit the Business Objects homepage.

Summary

A series of tutorial articles for how to get the most out of Crystal Xcelsius 4.5.

Authors: Loren Abdulezer (Evolving Technologies), Ryan Goodman, Chris Bryant (Business Objects), Thomas Gonzalez (BrightPoint Consulting), Michael Alexander (Crystal Xcelsius for Dummies), James Wakefield (Cortell NZ), David McAmis (Avantis Information Systems), David Harper (Investor Alternatives)

Company: (see individual bios below)

Created on: 01 November 2008

Author Bios

Loren Abdulezer is the CEO of Evolving Technologies Corporation (ETC), and author of the best-selling Excel Best Practices for Business. ETC, an Xcelsius Consulting Partner, is a technology consulting firm based in New York City. More information about Xcelsius can be found on Loren's web site: XcelsiusBestPractices.com. He can be reached at la@evolvingtech.com.

Michael Alexander is the author of several books including "Crystal Xcelsius for Dummies." He currently lives in Frisco, Texas where he runs DataPigTechnologies.com, providing training to beginning and intermediate users of Excel, Access, and Crystal Xcelsius.

James Wakefield is a Senior Consultant with Cortell in Wellington, New Zealand. James has completed a number of projects on performance and management reporting including the delivery of New Zealand Post's Value Scorecard. james.wakefield@cortellgroup.com

David McAmis is a BI Evangelist for Avantis Information Systems Pty Ltd, a successful IT consulting firm based in Sydney, Australia; as well as the chief editor for www.crystaldevelopersjournal.com, an independent publication written to help end-users and developers learn advanced techniques for making the most of the tools available from Business Objects.

David Harper is publisher of Bionic Turtle (www.bionicturtle.com), a test preparation service for the financial exam certification market. He is also Principal of Investor Alternatives, LLC, (www.investoralternatives.net) a firm that specializes in investment research, software sector coverage, and derivatives valuation. He is Contributing Editor of Investopedia Advisor (advisor.investopedia.com), a newsletter devoted to the early recognition of public companies that are likely to lead future markets. He is a Charted Financial Analyst (CFA) and Financial Risk Manager (FRM).

Table of Contents

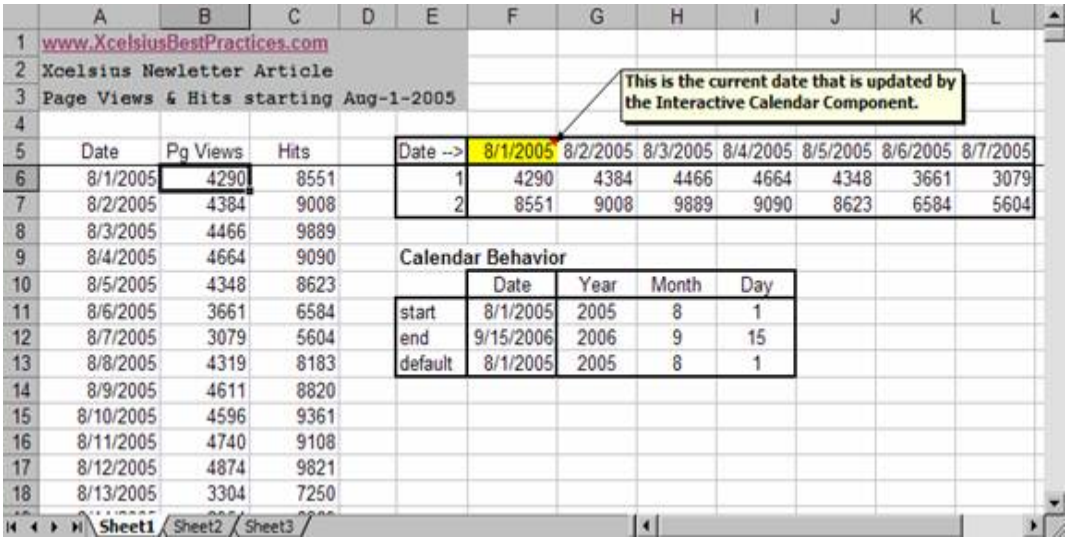
Creating Interactive Calendars in Xcelsius

by Loren Abdulezer, CEO, Evolving Technologies Corporation, an Xcelsius Consulting Partner

Last month I showed you how to do some clever things with Sliders. This month I want to show a simple but effective facility in Xcelsius. The component we'll be concentrating on is the Interactive Calendar. *(Please note: the Interactive Calendar component is only available on Xcelsius XL Professional and Enterprise XE products.)*

Setting up your spreadsheet for the Interactive Calendar

The Interactive Calendar component makes it easy to view data over a desired time period. The example I would like to use involves data on page views and hits to a web site over a range of dates. These appear in columns A, B, and C in Figure 1.



www.XcelsiusBestPractices.com
Xcelsius Newletter Article
Page Views & Hits starting Aug-1-2005

This is the current date that is updated by the Interactive Calendar Component.

Date	Pg Views	Hits
8/1/2005	4290	8551
8/2/2005	4384	9008
8/3/2005	4466	9889
8/4/2005	4664	9090
8/5/2005	4348	8623
8/6/2005	3661	6584
8/7/2005	3079	5604
8/8/2005	4319	8183
8/9/2005	4611	8820
8/10/2005	4596	9361
8/11/2005	4740	9108
8/12/2005	4874	9821
8/13/2005	3304	7250

Date -->	8/1/2005	8/2/2005	8/3/2005	8/4/2005	8/5/2005	8/6/2005	8/7/2005
1	4290	4384	4466	4664	4348	3661	3079
2	8551	9008	9889	9090	8623	6584	5604

Calendar Behavior

	Date	Year	Month	Day
start	8/1/2005	2005	8	1
end	9/15/2006	2006	9	15
default	8/1/2005	2005	8	1

Sheet1 Sheet2 Sheet3

It would be nice to view selections of this data over any given week. This is handled in the table at E5:L7 on the sample spreadsheet. The highlighted cell (located at F5 on the spreadsheet) is the beginning date for the range of dates you're interested in viewing. There is no formula for this cell, as its value is revised every time you click on a different date in the calendar.

All the dates appearing to its immediate right (cells G5:L5) are set by incrementing the selected date by 1.

The values for page views and hits are looked up using the Excel VLOOKUP function. Since the spreadsheet file is available at: http://www.xcelsiusbestpractices.com/landingZone/articleReprints.html , I'll allow you to go through all the formulas including VLOOKUP on your own.

Setting up an Interactive Calendar

In your Xcelsius work area, drag and drop the Interactive Calendar component onto the canvas. Double-click on it to open its Properties Panel, and set your Insert Source Data.

For the moment, I am going to hold off on doing more work on the calendar component. We'll add the finishing touches shortly.

Putting the Calendar Component to Work

Now it's time use the date selected by clicking on the calendar to show some information. Drop a Column Chart component onto the Xcelsius canvas. Go to its Properties Panel, and select the Date Range to be cells F5:L7.

While we haven't yet added the finishing touches, if you preview it you can generate Xcelsius charts that will allow you to visually sift through the data over any range of dates.

The Finishing Touches

Notice that I said "*any range of dates*". But, is that always a good thing? You may want to restrict the range of dates in your calendar component so that it does not, figuratively speaking, run off the road. You can do this by setting the Interactive Calendar's behavior from the Behavior tab in its Properties Panel. There are two ways of doing this. One of these is to set the calendar's behavior to a “hardwired” range of dates. The other is to have it read the limits directly off the spreadsheet .

In this case, I've done both of these techniques. I have set the start date to read the values directly from the spreadsheet, and I entered the end date by hand.

These are the basic concepts and techniques involved with Interactive Calendars. Like so many components you can use in Xcelsius, there is really nothing magical about how to work them, but their effects are almost magical.

For the full-article, visit
http://www.xcelsiusjournal.com/index.php?option=com_content&task=view&id=63&Itemid=2

Loren Abdulezer is the CEO of Evolving Technologies Corporation (ETC), and author of the best-selling Excel Best Practices for Business. ETC, an Xcelsius Consulting Partner, is a technology consulting firm based in New York City. More information about Xcelsius can be found on Loren’s web site: XcelsiusBestPractices.com. He can be reached at la@evolvingtech.com. *The files referenced in this article can be found in the Article Reprints section of: http://www.xcelsiusbestpractices.com*

Xcelsius Multi-Layer Dashboard in BusinessObjects Enterprise

By Ryan Goodman

Inside of BusinessObjects Enterprise (BOE), you can deploy a mult-layer dashboard with SSO enabled. You can also use this methodology to pass variables to the child SWF. In this case, you would not only be loading a Child SWF, but you would also be sending parameters to dictate the information to show up in the child SWF

There are 3 primary concepts used in this workflow:

1: Multi-Layer Dashboards: A multi-layer dashboard is defined as a parent Xcelsius SWF file that dynamically loads external SWFs inside a nested slideshow or image component. In our case we will pass a dynamic URL to a slideshow component inside of our parent SWF.

2: Flash Variables: A Flash Variable will load any variable from the framework for which an Xcelsius SWF is loaded (HTML, SWF, .NET, etc).
In our case will push the login token as a Flash Variable from the BOE instance (HTML) that the SWF resides in.

3: Concatenate a URL: In many instances you can dynamically concatenate variables and from within a model to dynamically generate a URL during runtime. This URL can be used to load or send parameterized queries, or launch reports.

Step by Step Instructions

Configuring the Child SWFs

1. Build and publish your child SWF files to the BOE repository.
2. Locate the "iDocID" for each child SWF. The easiest way I found to locate this ID, was to navigate to the BOE repository and load each SWF file inside of Infoview. I would then right click on the HTML off to the right hand side of the screen and click "View Source" to show the HTML code. I would then locate the iDocID.

Build the Excel File

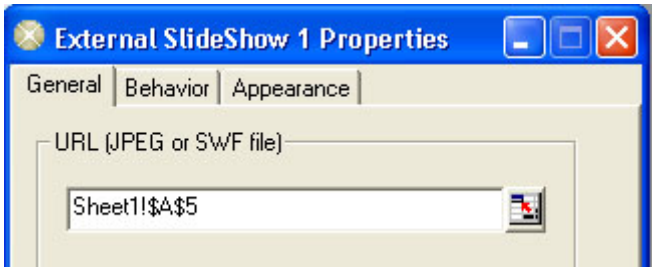
	A	B	C	D	E
1		**Token**			
2	**Flash Variable**				
3					
4	**Concatenate**				
5	documentDownload?iDocID=10348&sKind=Flash&CELogonToken=				
6					
7		**Label**	**iDocID**		
8	**Insert In**	Gross Sales	10348		
9					
10		**Label**	**iDocID**		
11		Gross Sales	10348		
12		Headcount	8934		
13		Regional Growth	8744		
14					
15					

3. Take all of the iDocIDs and load them into the Excel file. In my case I knew that a selector would be used toggle between my child SWFs. I reserved a space for my insert in row for my selector (yellow).
4. Create a space for your Login Token, which will be pushed to the model via Flash Variables (pink).
5. Concatenate the URL for the child SWF inside of BOE. This concatenated URL will be based on the following URL:

documentDownload?iDocID=10348&sKind=Flash&CELogonToken=VANPCHENG.PRODUCT.BUSINESSOBJECTS.COM%408757Jxmhe3URUJ8L4D8b8755JihGX0vShOZU9CPc

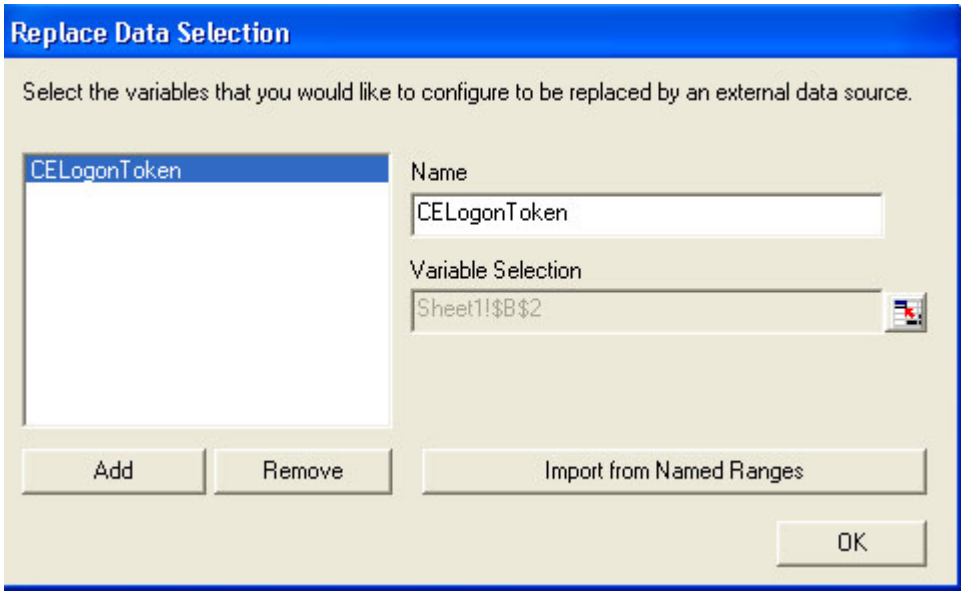
6. Save the Excel file and import into Xcelsius.

Build the Xcelsius Model



7. Insert a Slideshow Component, which is located in the components window within the Connectivity folder.
8. In the Slideshow properties window, bind the URL to the concatenated URL (A5).
9. Create the rest of your Xcelsius model. In our case I added my selector to allow us to toggle between child SWFs.

Configure the login token Flash Variable



10. Add the Flash Variable: Click on File>Export Settings.. Click on Define Variables
11. Add a variable and name it "CELogonToken". Then bind the value to the spreadsheet (B2). The name is extremely important to identify the variable inside of Infoview.
12. Save and publish your model to the repository.

Other Considerations

In this case we use the iDocID. Others have questioned this from a migration standpoint because this ID changes. If anyone has any ideas on other ways to call the child SWFs, please add comments or edit this document.

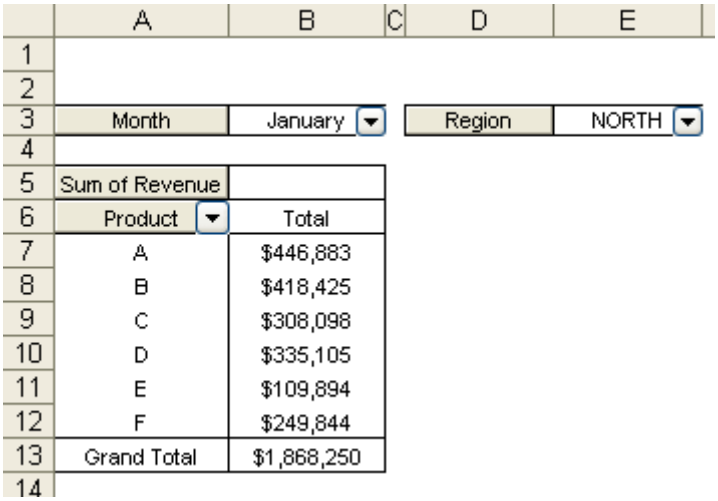
Simulating Pivot Table Functionality in Crystal Xcelsius

By Michael Alexander, author of "Crystal Xcelsius for Dummies"

Pivot Tables and Crystal Xcelsius: What's the deal?

For those of you who are not familiar with Excel pivot tables, let me first explain what a pivot table is. A pivot table is an analysis tool that allows you to create an interactive view of your dataset. With a pivot table report, you can quickly and easily categorize your data into groups and summarize large amounts of data into meaningful information

Figure 1 demonstrates a pivot table that summarizes revenue for each product in a given month and region. As you can see here, this pivot table is set to show product revenue for the North region in January. Keep in mind that this pivot table is not merely selecting one row of data and presenting it. It is actually aggregating all the rows that meet the month and region criterion and presenting the sum value of those rows. That is the power of pivot tables.



	A	B	C	D	E
1					
2					
3	Month	January		Region	NORTH
4					
5	Sum of Revenue				
6	Product	Total			
7	A	$446,883			
8	B	$418,425			
9	C	$308,098			
10	D	$335,105			
11	E	$109,894			
12	F	$249,844			
13	Grand Total	$1,868,250			
14					

Figure 1: Pivot tables allow you categorize and aggregate large amounts of data.

As you may well know, Crystal Xcelsius does not support the use of Excel pivot tables. That is to say, you can not use pivot tables or their functionality in your visual models. The primary reason for this is due to the way pivot tables work.

When you work with a pivot table on your spreadsheet, it may feel as though you are working with an object that is connected to your dataset. In fact, you are actually working with a disconnected cache of data called a pivot cache. The pivot cache is essentially a snapshot of your dataset stored in memory on your local system. This is why your must refresh your pivot table when your dataset changes; to take another snapshot. The bottom line is that when look at a pivot table on your spreadsheet, you are merely seeing values that have been returned by the pivot cache.

Now stay with me on this thought. When Crystal Xcelsius imports your Excel model, it captures all of the values and formulas on your spreadsheet. Since what you perceive to be a pivot table is actually a set of values that have been returned by the pivot cache, Crystal Xcelsius only imports those values. Crystal Xcelsius can not reach into your system and pull in the pivot cache that makes up the core of your pivot table. This is essentially why you can not use pivot tables in your visual models.

Now before you get that defeated look on your face, you must know that pivot tables are not the end-all-be-all when it comes to data analysis. There are a handful of Excel functions that, if used correctly, can help simulate pivot table functionality. In this article we will explore one of these functions and use it to create a dashboard that works like a pivot table!

Note: To follow along with this article, you will need the associated source files. The source files for this article can be found here.

Introducing the SUMIF function

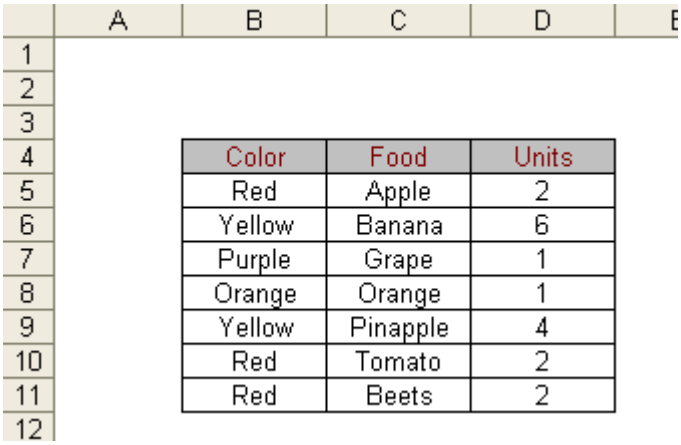
Excel's SUMIF function allows you to sum up several rows in a range if a value in those rows meets a criterion you specify. In English, this means the SUMIF function tells Excel to add up the rows in a range of cells only if a condition is true.

The SUMIF function requires three arguments in order to work properly:

- The range to evaluate
- The criteria to check for
- The range to sum up

The syntax will look like this: SUMIF(range, criteria, sum_range)

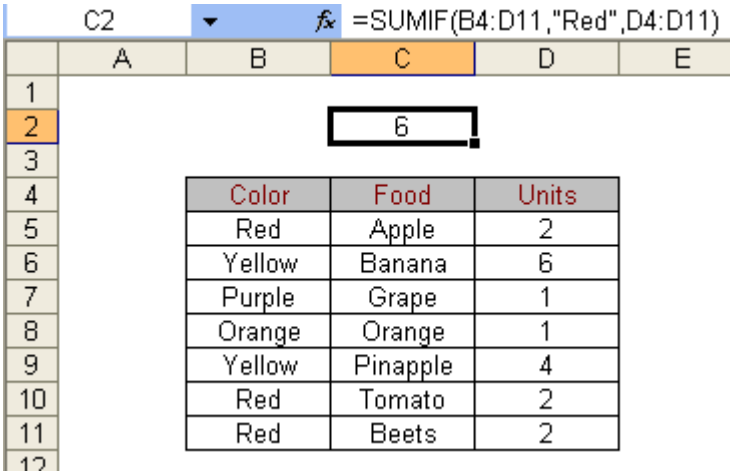
Let's walk through a simple example of using the SUMIF function. In Figure 2, we have a table that shows us foods, colors, and units.



Color	Food	Units
Red	Apple	2
Yellow	Banana	6
Purple	Grape	1
Orange	Orange	1
Yellow	Pinapple	4
Red	Tomato	2
Red	Beets	2

Figure 2: Basic table of foods, colors, and units.

Suppose we wanted to find out how many units of red foods we have. We could use the following SUMIF formula: =SUMIF(B4:D11,"Red",D4:D11). As you can see in Figure 3, giving the SUMIF formula the criteria "Red" adds up the units for all the red foods.



Color	Food	Units
Red	Apple	2
Yellow	Banana	6
Purple	Grape	1
Orange	Orange	1
Yellow	Pinapple	4
Red	Tomato	2
Red	Beets	2

Figure 3: You can explicitly enter your criteria within the SUMIF formula.

Note: In a SUMIF formula, your criteria must come from the first column in the range.

Interestingly enough, you don't have to explicitly enter the criteria as in Figure 3. You can define your criteria by referencing a cell. In Figure 4, we reference cell A6 to supply the criteria value for the SUMIF formula. This essentially tells the formula to add up all the rows that have the color Yellow.

C2 =SUMIF(B4:D11,B6,D4:D11)

	A	B	C	D	E
1					
2			10		
3					
4		Color	Food	Units	
5		Red	Apple	2	
6		Yellow	Banana	6	
7		Purple	Grape	1	
8		Orange	Orange	1	
9		Yellow	Pinapple	4	
10		Red	Tomato	2	
11		Red	Beets	2	
12					

Figure 4: You can also use a cell reference to supply a criteria value.

Now that we have a basic understanding of the SUMIF function, let's look at how we can use it to create a dashboard that works like a pivot table.

Simulating a Pivot Table with the SUMIF function

We start with a basic table, shown here in Figure 5, that gives us the revenue and units sold by product for each region in our organization. This is further broken down by month.

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						
8						
9		Region	Month	Product	Revenue	Units
10		MIDWEST	April	C	$3,929.18	39
11		MIDWEST	April	D	$4,335.05	31
12		MIDWEST	April	E	$4,812.36	28
13		MIDWEST	April	F	$1,350.55	5
14		MIDWEST	April	A	$2,158.11	11

Figure 5: Basic Table that shows revenue and units by product, region and month.

Next, we add a column that will give us a concatenated value that identifies the region, month and product for that row. This is the value we will use as the criteria in our SUMIF formulas. The concatenated value shown in Figure 6 was created by using the following formula: =B10&C10&D10.

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						
8						
9	Key	Region	Month	Product	Revenue	Units
10	MIDWESTAprilC	MIDWEST	April	C	$3,929.18	39
11	MIDWESTAprilD	MIDWEST	April	D	$4,335.05	31
12	MIDWESTAprilE	MIDWEST	April	E	$4,812.36	28
13	MIDWESTAprilF	MIDWEST	April	F	$1,350.55	5
14	MIDWESTAprilA	MIDWEST	April	A	$2,158.11	11

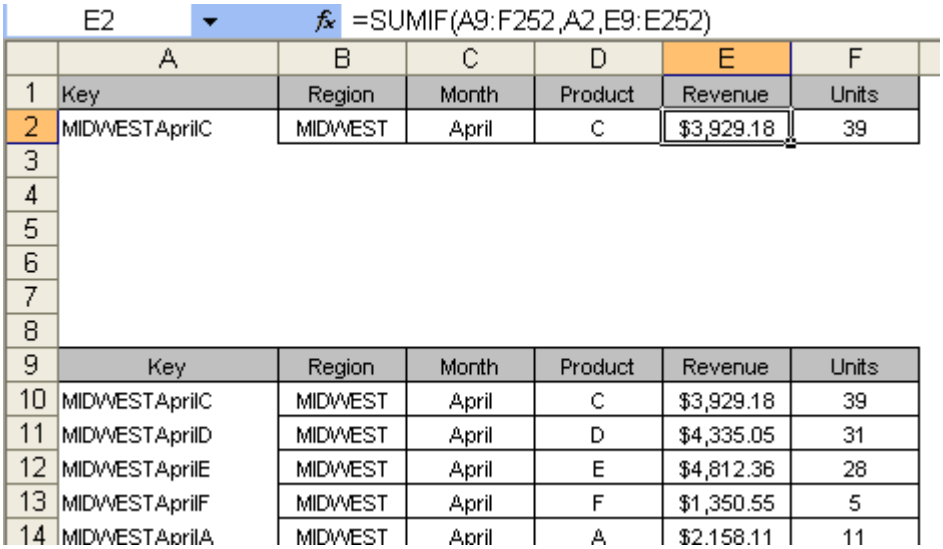
Figure 6: Create a concatenated value to use as your criteria in your SUMIF formulas.

Now we copy the first row of our table (plus the headers) to the top of the Excel model. This is the area Crystal Xcelsius will work with. At this point your worksheet should look similar to Figure 7.

	A	B	C	D	E	F
1	Key	Region	Month	Product	Revenue	Units
2	MIDWESTAprilC	MIDWEST	April	C	$3,929.18	39
3						
4						
5						
6						
7						
8						
9	Key	Region	Month	Product	Revenue	Units
10	MIDWESTAprilC	MIDWEST	April	C	$3,929.18	39
11	MIDWESTAprilD	MIDWEST	April	D	$4,335.05	31
12	MIDWESTAprilE	MIDWEST	April	E	$4,812.36	28
13	MIDWESTAprilF	MIDWEST	April	F	$1,350.55	5
14	MIDWESTAprilA	MIDWEST	April	A	$2,158.11	11

Figure 7: Copy first row of data to the top of the sheet. Note that the Key column is still a formula.

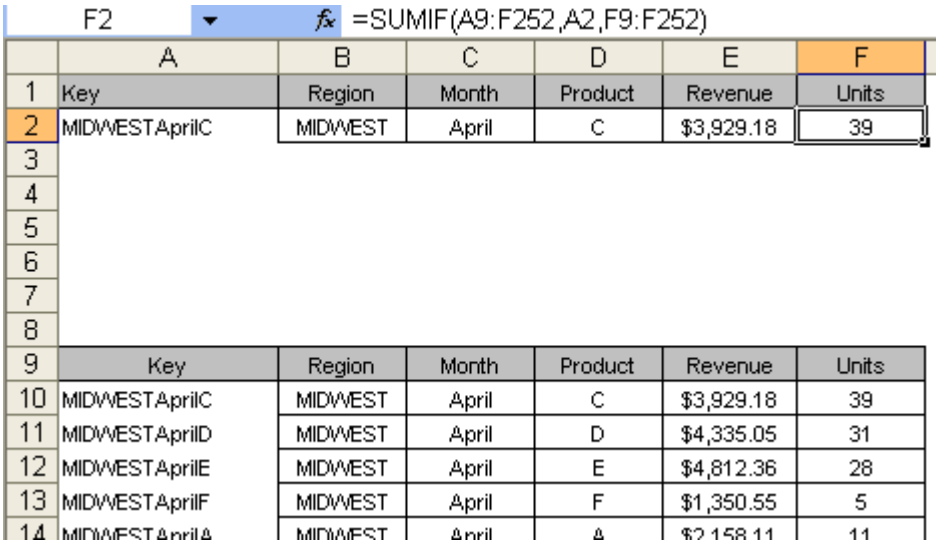
From here, we can replace the Revenue value in cell E2 with the following SUMIF formula: =SUMIF(A9:F252,A2,E9:E252). This formula tells Excel to return the sum of the "Revenue" column from the table in A9:F252 where the first column equals the value in cell A2. As you can see in Figure 8, the concatenated "Key" field we created is doing its job, serving as a field that Excel uses to categorize and sum.



	A	B	C	D	E	F
1	Key	Region	Month	Product	Revenue	Units
2	MIDWESTAprilC	MIDWEST	April	C	$3,929.18	39
3						
4						
5						
6						
7						
8						
9	Key	Region	Month	Product	Revenue	Units
10	MIDWESTAprilC	MIDWEST	April	C	$3,929.18	39
11	MIDWESTAprilD	MIDWEST	April	D	$4,335.05	31
12	MIDWESTAprilE	MIDWEST	April	E	$4,812.36	28
13	MIDWESTAprilF	MIDWEST	April	F	$1,350.55	5
14	MIDWESTAprilA	MIDWEST	April	A	$2,158.11	11

Figure 8: Create the first SUMIF formula to return aggregate revenues.

Do the same thing for Units by replacing the hard-coded value with the formula shown here in Figure 9.



	A	B	C	D	E	F
1	Key	Region	Month	Product	Revenue	Units
2	MIDWESTAprilC	MIDWEST	April	C	$3,929.18	39
3						
4						
5						
6						
7						
8						
9	Key	Region	Month	Product	Revenue	Units
10	MIDWESTAprilC	MIDWEST	April	C	$3,929.18	39
11	MIDWESTAprilD	MIDWEST	April	D	$4,335.05	31
12	MIDWESTAprilE	MIDWEST	April	E	$4,812.36	28
13	MIDWESTAprilF	MIDWEST	April	F	$1,350.55	5
14	MIDWESTAprilA	MIDWEST	April	A	$2,158.11	11

Figure 9: Create a SUMIF formula to return aggregate units.

Let's stop here and take a moment to think about what we've built. Each time we change the region, month or product, we get new aggregate totals! So the idea is to tie region and month to Combo Box components so a user can select which aggregate they would like to see.

	A	B	C	D	E	F
1	Key	Region	Month	Product	Revenue	Units
2	NorthJanuaryA	North	January	A	$446,883.08	2433
3						
4						
5						
6						
7						
8						
9	Key	Region	Month	Product	Revenue	Units
10	MIDWESTAprilC	MIDWEST	April	C	$3,929.18	39
11	MIDWESTAprilD	MIDWEST	April	D	$4,335.05	31
12	MIDWESTAprilE	MIDWEST	April	E	$4,812.36	28
13	MIDWESTAprilF	MIDWEST	April	F	$1,350.55	5
14	MIDWESTAprilA	MIDWEST	April	A	$2,158.11	11

Figure 10: The cells in yellow will be tied to combo box components, allowing users to interactively change the aggregate totals for the products.

Next, we'll hard-code the product names so they are in fixed positions on our dashboard. Then we will create the criteria key for each product by using the month and region values that come from the Combo Box components. Figure 11 demonstrates the formula that will do this.

A3 fx =B2&C2&D3

	A	B	C	D	E	F
1	Key	Region	Month	Product	Revenue	Units
2	NorthJanuaryA	North	January	A	$446,883.08	2433
3	NorthJanuaryB			B		
4				C		
5				D		
6				E		
7				F		
8						
9	Key	Region	Month	Product	Revenue	Units
10	MIDWESTAprilC	MIDWEST	April	C	$3,929.18	39
11	MIDWESTAprilD	MIDWEST	April	D	$4,335.05	31
12	MIDWESTAprilE	MIDWEST	April	E	$4,812.36	28
13	MIDWESTAprilF	MIDWEST	April	F	$1,350.55	5
14	MIDWESTAprilA	MIDWEST	April	A	$2,158.11	11

Figure 11: Create a new row for each product and fill the criteria key by using the formula shown here.

All there is left to do is fill in the table by copying the SUMIF formulas down. In the end, you will have a table that looks similar to the one in Figure 12.

	A	B	C	D	E	F
1	Key	Region	Month	Product	Revenue	Units
2	NorthJanuaryA	North	January	A	$446,883.08	2433
3	NorthJanuaryB			B	$418,425.25	1007
4	NorthJanuaryC			C	$308,098.27	2461
5	NorthJanuaryD			D	$335,105.04	2189
6	NorthJanuaryE			E	$109,893.93	917
7	NorthJanuaryF			F	$249,844.32	1315
8						
9	Key	Region	Month	Product	Revenue	Units
10	MIDWESTAprilC	MIDWEST	April	C	$3,929.18	39
11	MIDWESTAprilD	MIDWEST	April	D	$4,335.05	31
12	MIDWESTAprilE	MIDWEST	April	E	$4,812.36	28
13	MIDWESTAprilF	MIDWEST	April	F	$1,350.55	5
14	MIDWESTAprilA	MIDWEST	April	A	$2,158.11	11

Figure 12: Copy the formulas down to fill the table.

Warning: *Make sure the cell references in your formulas don't shift as you copy them down. You can use absolute references to avoid this problem.*

I've taken this Excel model and built the dashboard in Figure 13 on top of it. As you can see, the dashboard has a pivot table feel to it. With each change in region and market, Crystal Xcelsius recalculates the aggregate totals shown in the grid.

Tip: *I have created a step-by-step video on this very technique.* *Click here to watch the video!*

As you think about what you have just learned, consider this; the SUMIF function is not the only function that can be used for these types of analyses. Crystal Xcelsius supports a whole range of functions that enable you to perform aggregate calculations; DSUM, DCOUNT, DAVERAGE and SUMPRODUCT are just a few. Familiarizing yourself with these supported Excel functions will open a whole new level of possibilities, allowing you to go far beyond the perceived limitations of Crystal Xcelsius.

Using Flash Var with Crystal Xcelsius

By Chris Bryant, Product Manager, Business Objects

Flash Vars

One of the most frequently asked questions for Crystal Xcelsius customers that are integrating their dashboards with a portal or web sites is: "How can I pass parameters to the Dashboard when it loads?" This document will take a look at a feature in Crystal Xcelsius called Flash Variables. This feature allows Dashboard Designers to define ranges of information in Crystal Xcelsius that are to be replaced on load from an html page. The html page will set the Flash Vars variable, call the SWF, and push the ranges of information from the html page to the SWF file. This functionality will allow a portal or web site to pass user information, configuration parameters, or pre-defined criteria to the SWF.

This document contains three sections:

1. A description of how Flash Vars work.
2. A sample scenario of when one would use Flash Vars with Crystal Xcelsius.
3. A step-by-step directions on how to define Flash Vars within a dashboard.

Download the source file referenced in this article here.

How do Flash Vars Work

Flash Vars can be used to pass any piece of information to the SWF file upon load. This differs from a data connection via XML that allows for accessing information only after the SWF has been loaded. The benefit of using Flash Vars is the ability to pass parameters to the SWF file that will be used to either configure the dashboard, or used as criteria and login information for the queries back to the data source.

The Dashboard designer defines the information that they want replaced with outside parameters by defining a variable and selecting a range of cells for that variable. This information is passed to the SWF file from an html page by setting the Flash Vars variable in the call to load the SWF. The information can be passed in two different formats, CSV and XML. If XML is used then the Crystal Xcelsius standard Row, Column XML structure needs to be used, hence CSV is generally easier to work with.

Example of how Flash Vars can be used

This section will discuss a sample scenario for using the Flash Vars functionality.

When integrating a Crystal Xcelsius designed dashboard into a portal environment that makes real-time connections back to the database, Flash Vars can provide for a seamless and easy-to-implement solution. The portal environment already handles authentication against the security database, as well as specifies which users have access to the various applications inside the shared environment. Using the same authentication techniques, you can ensure that only specific users have access to your dashboard. When the user opens the dashboard, the SWF will need to connect to the database to pull back the data that confirms the user has rights to access and populate the dashboard with the new data. Using Flash Vars, a web developer can easily capture the user's login information from the portal environment, and then pass the credentials to the SWF file upon load. Once the SWF file has the proper credentials it will then pass the information back to the database to run a query. You can also expand upon this idea, and run a query against the database that returns data specific to each individual user.

Example of setting Flash Variable in the html file

Here is a snippet of the format of the html used to call a SWF and to pass the parameters. This example uses the CSV format for the Flash Vars The Flash Vars variable must be set in 2 locations (both highlighted

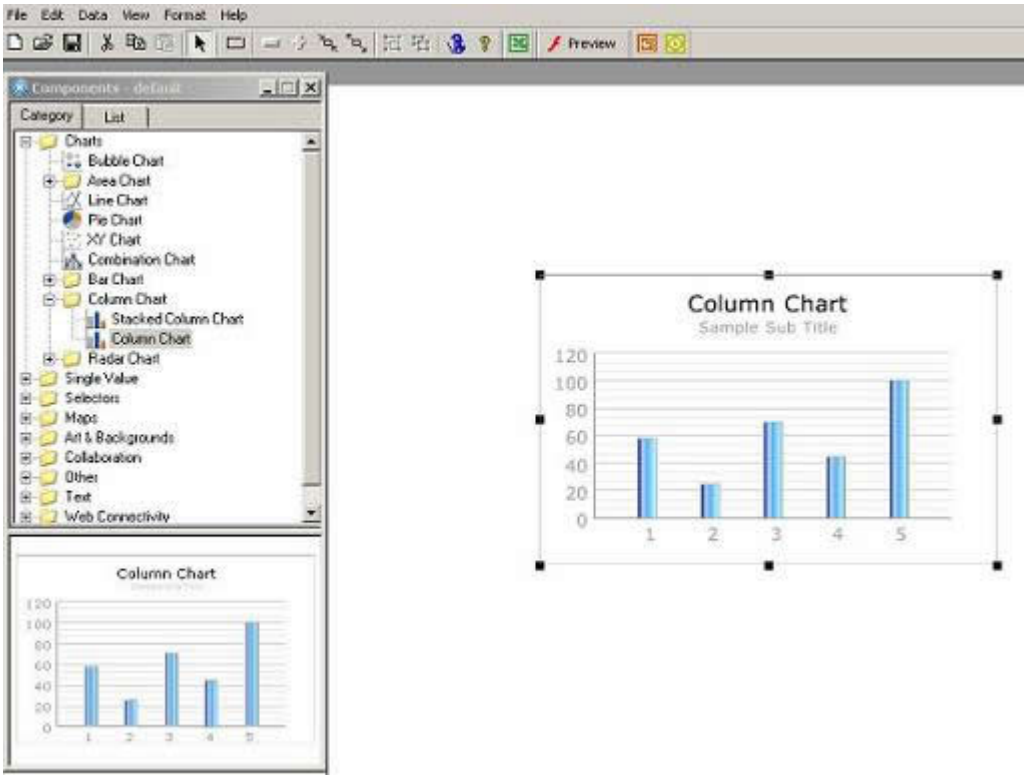
in the following example) the first is under PARAM and the second is in the EMBED tag. The EMBED tag is used for Netscape browsers while the PARAM section is used by all other browsers.

```
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/
swflash.cab#version=6,0,40,0" WIDTH="1231" HEIGHT="647" id="myMovieName">
<PARAM NAME=FlashVars VALUE="Range0=3.000000">
<PARAM NAME=movie VALUE="flashvars.swf">
<PARAM NAME=quality VALUE=high>
<PARAM NAME=bgcolor VALUE="#FFFFFF">
<EMBED src="flashvars.swf" quality=high bgcolor="#FFFFFF" WIDTH="1231"
HEIGHT="647" FlashVars="Range0=3.000000"
NAME="myMovieName" ALIGN="" TYPE="application/x-shockwave-flash"
PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer">
</EMBED>
</OBJECT>
```

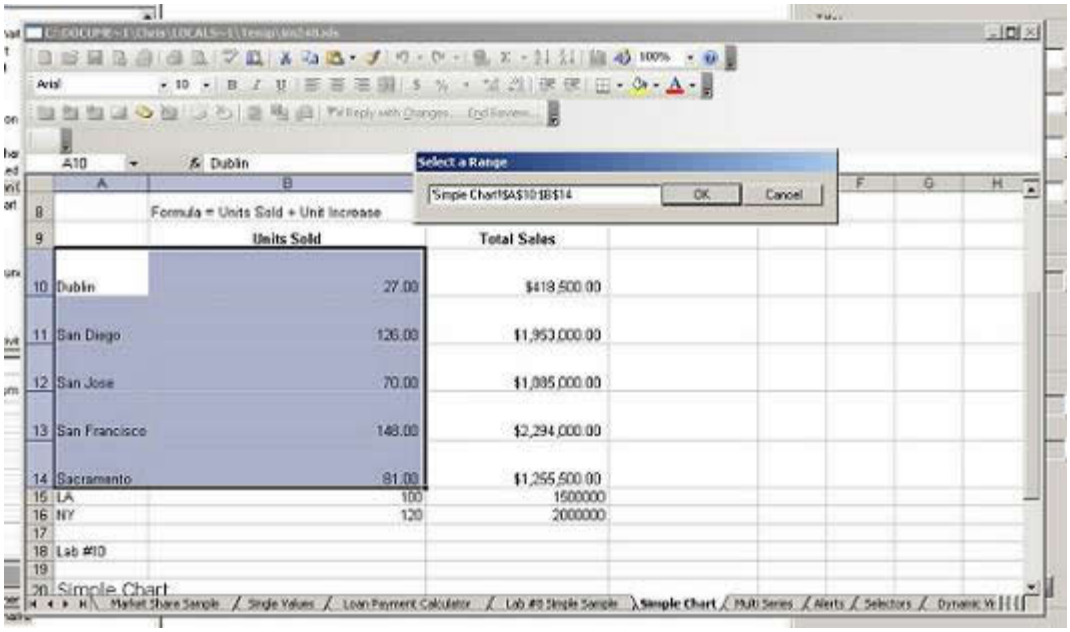
Step-by-Step directions to define Flash Vars

This section is going to describe the steps necessary to define Flash Vars in your Dashboard. We will use the Crystal Xcelsius Sample Models.xls spreadsheet.

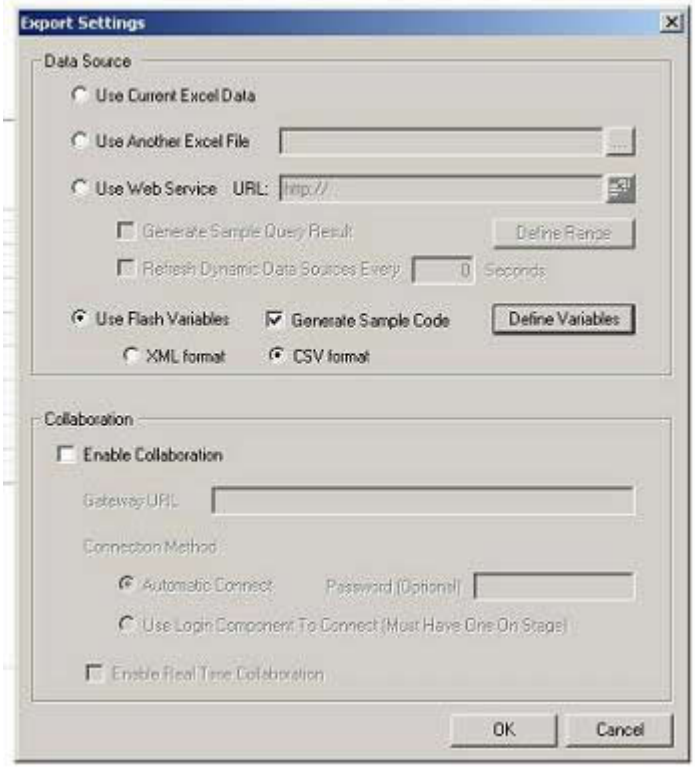
1. Open Crystal Xcelsius and press the excel icon in the toolbar to import in the sample spreadsheet. Browse to the Crystal Xcelsius Sample Models.xls file and click ok to import.
2. Select a Column chart from the component dialog box and place it in the middle of the canvas.



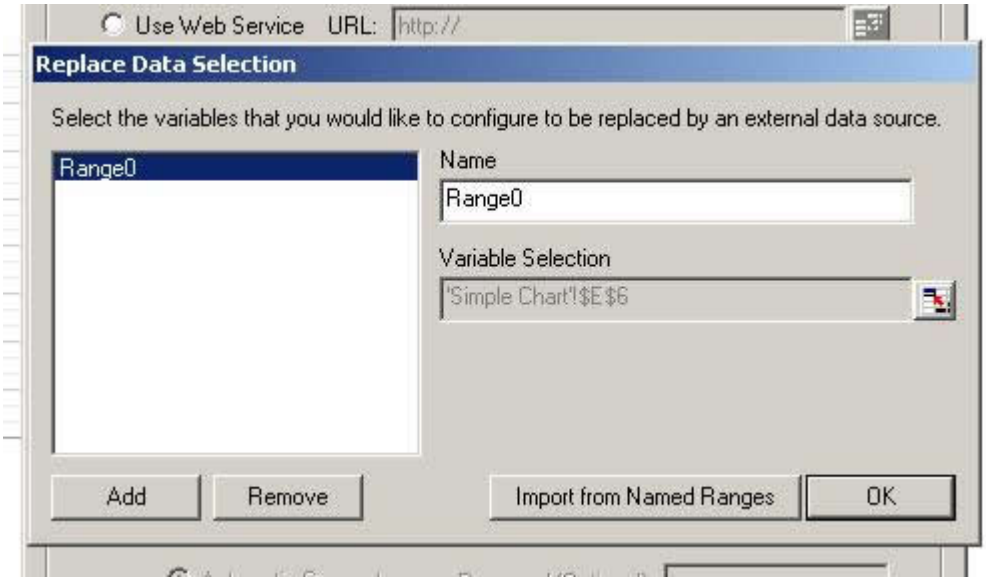
3. We will need to define some data to show on the chart. Open the properties dialog for the column chart and select cells 'Simple Chart'!A10:B14 to be displayed in the chart.



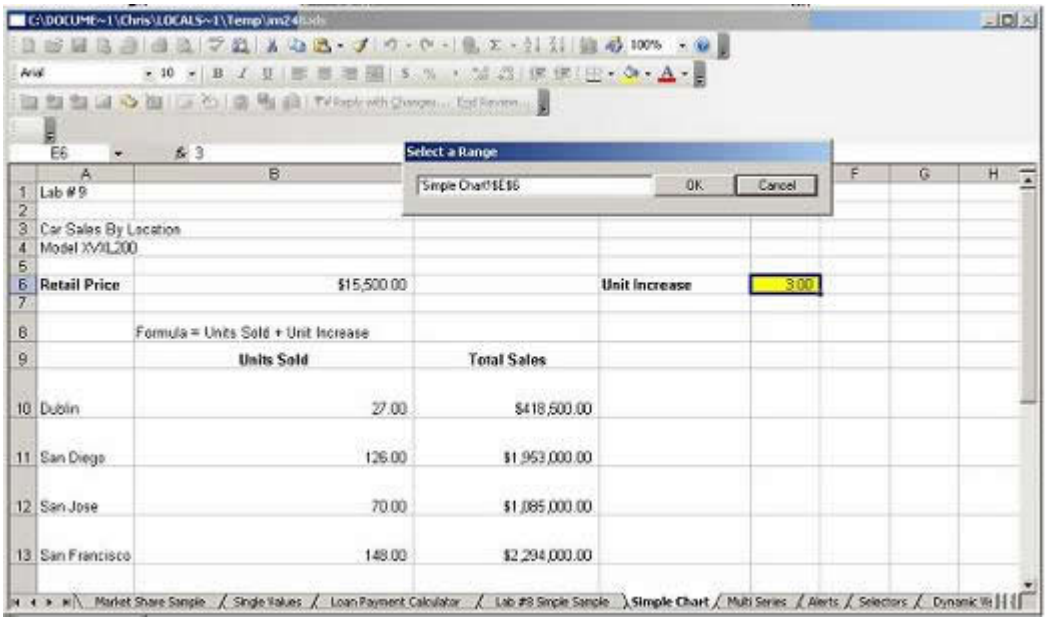
4. Next we need to define the data that we want replaced by the Flash Vars.
5. Go to the File Menu and select Export Settings.
6. Select the Use Flash Variables Radio Button and set the format to CSV.
7. Next select the check box for Generate Sample Code, this will generate an html file containing the code needed to call the SWF and set the Flash Vars.



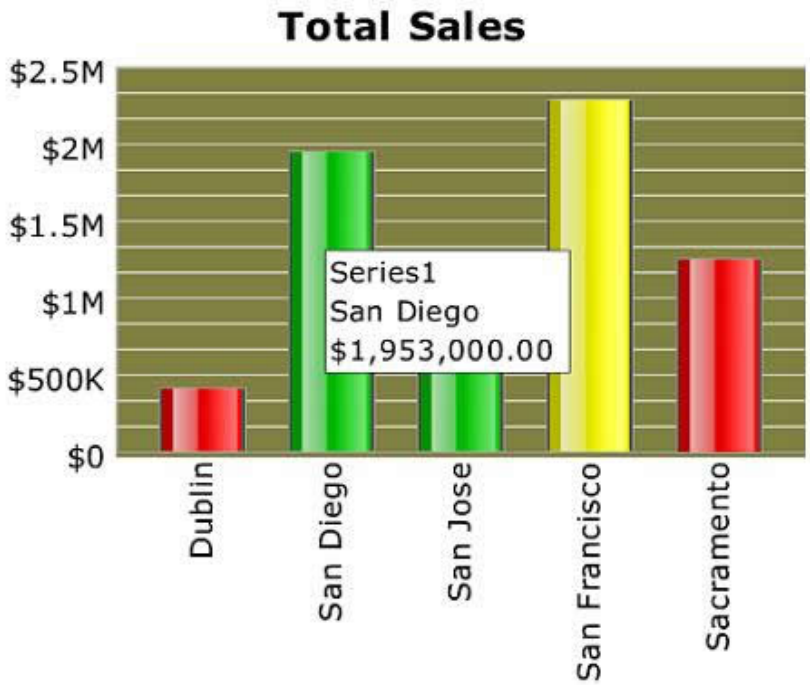
8. The next step is to define the range of data to be set by the Flash Vars.
9. Click on the Define Variables button to open the Replace Data Selection dialog box.
10. Click Add to add a new range, we can leave it called Range0 for this example.



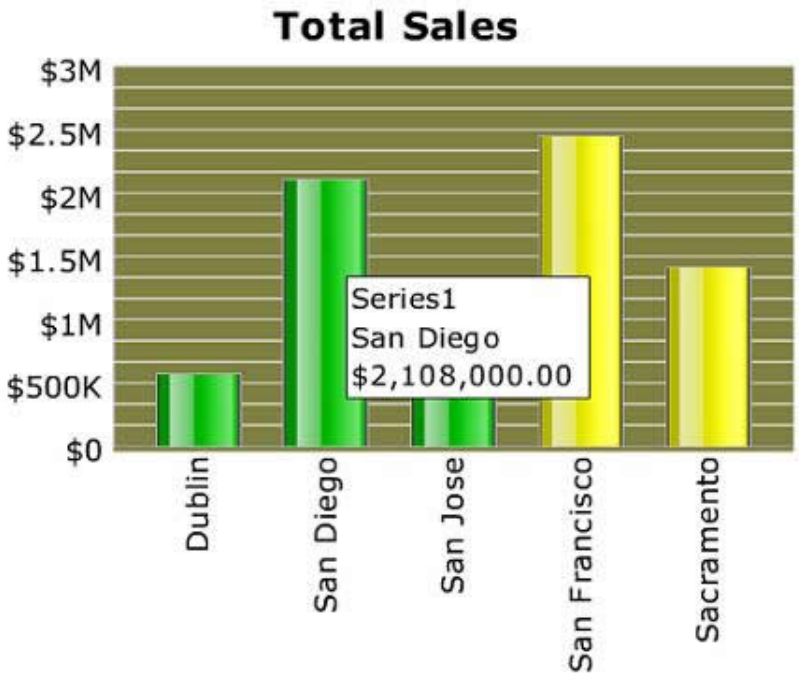
11. Click the **Variable Selection** button to define the cells for Flash Vars.
12. Select the 3.00 value in Cell E6 on the **Simple Chart** tab. Changing this value will change the data that we selected for our Column Chart.
13. Close the **Export Settings** dialog box.



14. We have now defined a Flash Variable. Let's export our SWF so we can test how this works. Go to the File Menu and Select Export->Macromedia Flash (SWF).
15. Save the SWF to any location, this example will be saving to the desktop as FlashVar.swf.
16. An html file (FlashVars.html) was generated in the same location as the swf. Double-click the html file which will open the swf. Make note of the values in the column chart. Now let's change our Flash Variable and see the effect.



17. Start by opening the html file in notepad. We need to change the Flash variables in the 2 spots mentioned in the Example of setting Flash Variable in the html file section above (PARAM, and EMBED tags). Change the value 3.00000 to 13.00000. Save the html file and open it again. We can see that all the values in our chart have now changed. Now open the SWF separately and compare the difference. By opening the SWF and not the html, the Flash Variable is not passed through, which is why we see the default values.



By integrating Flash Vars into your Crystal Xcelsius dashboards, you can create an entirely new set of security and data connectivity features that are not present "out-of-the-box." This level of functionality can be extremely useful, and in some cases necessary, when deploying a connected dashboard into a shared enterprise portal environment.

Connecting Crystal Xcelsius Dashboards to OLAP Data Sources using MDX, ASP and XML

By James Wakefield, Senior Consultant, Cortell New Zealand Ltd.

You might have already read the Crystal Xcelsius TechTips: Connectivity Using an XML Data Button and seen how you can connect your Crystal Xcelsius dashboard to live data using the following architecture:

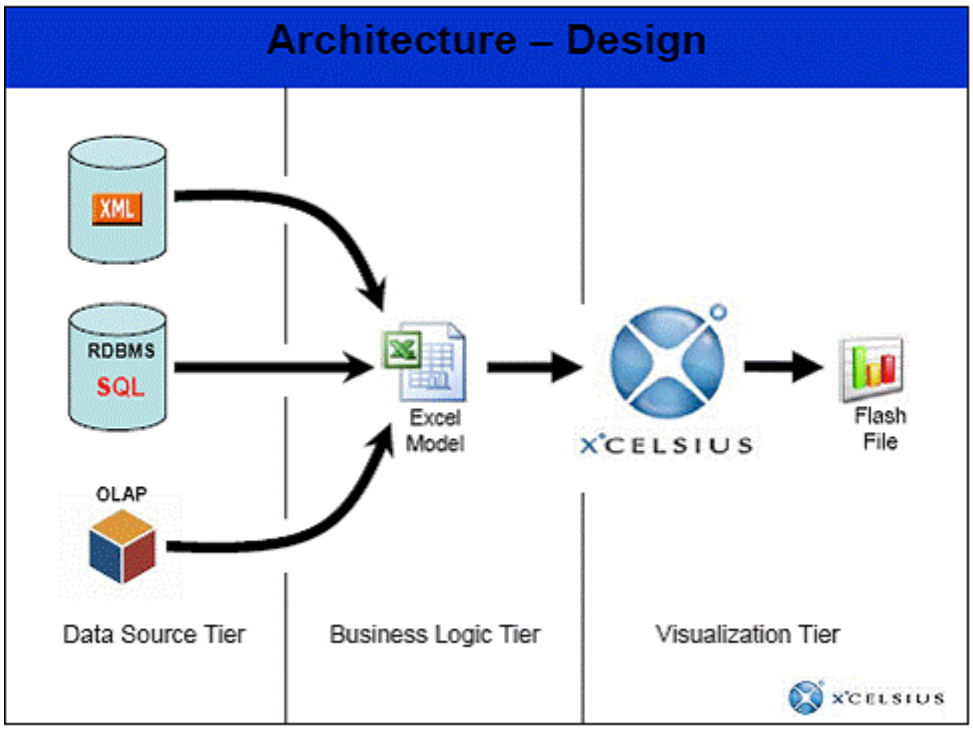


Figure 1

The scripts provided in that document concentrate on returning relational data via SQL statements, ADO, ASP and XML. Many OLAP databases accept MDX statements as a way of returning data from their cubes, and this article will show you how to connect to real-time OLAP data via MDX from your Crystal Xcelsius dashboard. Our script will return OLAP data via MDX statements, ADOMD, ASP and XML.

OLAP

The term OLAP stands for 'On-Line Analytical Processing'. Many databases are designated as OLTP, which stands for 'On-Line Transaction Processing.' OLTP databases record transactional information in tables. Whereas typically OLAP databases store information within cubes, and can hold transactional or summarized data. By storing data within cubes, OLAP databases allow quick and flexible, multi-dimensional views of data. This data can be sliced-and-diced to produce any type of report. A quick overview between the two types of databases can be seen in the following table:

Database Attribute	OLTP	OLAP
Real time Data	√	x
Detailed reporting	√	x
Speed	x	√
User Friendliness	x	√
Rich analytical capability	x	√

This is not to say that an OLAP database replaces an OLTP database. OLAP databases use OLTP databases as one of their data sources, and so they complement each other. OLTP databases will help an organization capture the transactional information that is important for their business operations; where OLAP databases allow managers and executives to effectively analyze the data and improve decision making within the organization.

Crystal Xcelsius allows companies to create highly visual and interactive dashboards. Executives typically do not need to see dashboards of transactional information, but rather of summarized data they can easily analyze to make effective management decisions. OLAP databases are also the ideal place to store Key Performance Indicators (KPIs), budgets and operational analytics. This means that the combination of an OLAP database plus Crystal Xcelsius can be a great Business Intelligence package for many companies.

MDX Statements

Multi-dimensional Expressions (MDX) is a syntax that supports the definition and manipulation of multi-dimensional objects and data. MDX is similar in many ways to the SQL syntax—with each query requiring a SELECT clause, a FROM clause being the name of the cube, and a WHERE clause for filtering. These and other keywords provide the tools used to extract specific portions of data from a cube, for analysis and reporting. Since MDX is designed specifically for returning data from OLAP databases, it is the most powerful and flexible way to present OLAP data to Crystal Xcelsius. Crystal Xcelsius just requires that it receives data in the specified XML format.

ADOMD

We will use ADOMD to provide the connection to our OLAP databases. ADOMD is based on the OLE DB for OLAP specification, and provides a set of COM objects that can be used to manipulate multi-dimensional data. ADO is similar to ADOMD, except ADO is designed for returning data from relational data in the form of tabular views. ADOMD is designed around hierarchies to return multi-dimensional data.

The Crystal Xcelsius Dashboard

As with any Crystal Xcelsius dashboard, the first place to start is by importing an Excel workbook into your Crystal Xcelsius work area. After you have created the dashboard layout you wish, add an XML Data Button to your dashboard and double-click to set its properties.

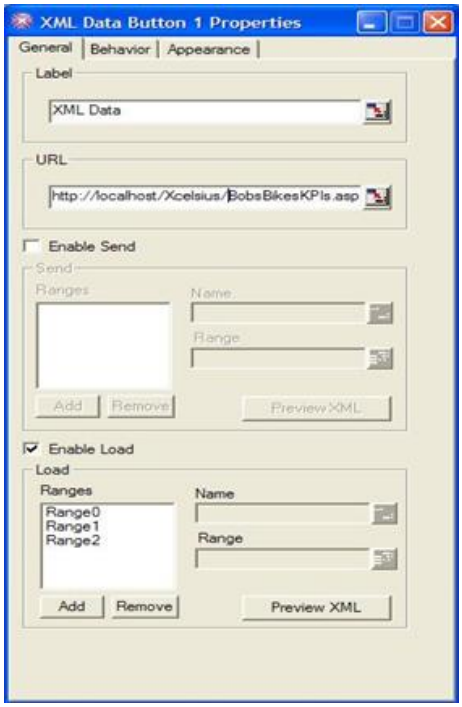


Figure 2

Un-check the Enable Send box and check the Enable Load box, instead. Add as many ranges as you need, but you will need one MDX statement per range. These ranges are typically the tables within your embedded Excel spreadsheet, and you will need to highlight the ranges within the spreadsheet as seen below:

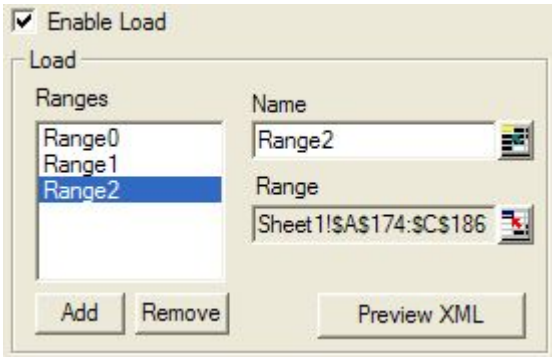


Figure 3

The ASP page must return the data from the MDX statement in the exact format outlined in the Preview XML button. The thing to be careful of here is to take note of the names that you assign to ranges, as these have to be included in the XML document. Below is an example of the XML format that Crystal Xcelsius might require.

ASP Page

Now, we need to create an ASP page that will connect to the OLAP database via ADOMD, issue an MDX statement and return the data in the XML format that Crystal Xcelsius requires.

Range2 in our Crystal Xcelsius project referred to Sheet1!$A$174:$C$186 in the Excel Workbook. To convert this into an MDX statement, we just need to specify the elements that are on the columns and rows, and then anything else becomes the filter criteria.

As such, the following table:

169	CUBE:	tm1serv:Profit and Loss Company 01 GBP		
170	scenario:	Actual		
171	group:	Total Group		
172				
173		**Gross Profit %**	**Net Profit %**	
174	**Jan**	46.05	-8.06	
175	**Feb**	46.12	27.31	
176	**Mar**	46.00	29.75	
177	**Apr**	45.96	26.42	
178	**May**	43.50	26.29	
179	**Jun**	43.47	23.59	
180	**Jul**	47.23	39.22	
181	**Aug**	47.20	38.50	
182	**Sep**	47.16	38.95	
183	**Oct**	47.13	38.99	
184	**Nov**	47.10	38.67	
185	**Dec**	47.06	36.63	

Figure 4

converts into the following MDX statement:

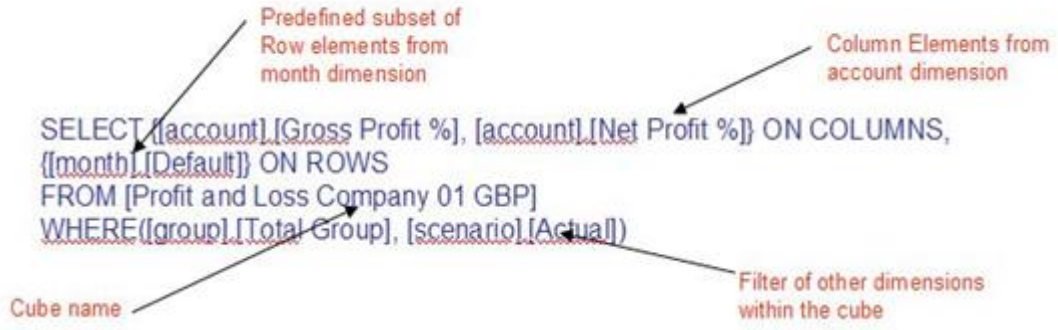


Figure 5

The MDX data needs to be converted into XML in order for Crystal Xcelsius to read it. To do this, we use a variable to hold the full XML string and keep on concatenating the rest of the data.

```
    sXML = "<data>"     sXML = sXML & "<variable name=" & Chr(34) & "Range0" &
Chr(34) & ">"     sXML = sXML & Mdx2Xml(strSource0)     sXML = sXML & "</variable>"
sXML = sXML & "<variable name=" & Chr(34) & "Range1" & Chr(34) & ">"     sXML = sXML
& Mdx2Xml(strSource1)     sXML = sXML & "</variable>"     sXML = sXML & "<variable
name=" & Chr(34) & "Range2" & Chr(34) & ">"     sXML = sXML & Mdx2Xml(strSource2)
sXML = sXML & "</variable>"     sXML = sXML & "</data>"
```

To create a connection to our OLAP database, we setup a connection via ADMOD. The following code creates two variables: the Catalog variable handles the connection to the database, and the CellSet variable handles the return of data via MDX.

```
Set cat = Server.CreateObject("ADOMD.Catalog") Set cst =
Server.CreateObject("ADOMD.CellSet")
```

The Catalog ActiveConnection property will need adjusting, depending upon the OLAP database that is being connected to.

Provider is the name of the OLE DB Provider:

```
TM1 = TM1OLAP, MSAS = MSOLAP, SAPBW = Mdrmsap
```

Location is the IP address of the server where your database resides.

Initial Catalog is the name of the database or Catalog being connected to.

```
cat.ActiveConnection = "Provider=TM1OLAP;Location=127.0.0.1;Initial
Catalog=tm1serv;User ID=admin;Password=apple;"  cst.Source = strSource Set
cst.ActiveConnection = cat.ActiveConnection cst.Open
```

It is always a good best-practice to close any connections that you create, and destroy any variables at the end of the script.

```
cst.Close  Set cst = Nothing Set cat = Nothing
```

Because ADOMD allows the return of hierarchical data, we can return the elements from each axis of the CellSet, and also the data, by looping round the dimension counts.

```
    intDC0 = cst.Axes(0).DimensionCount - 1     intDC1 = cst.Axes(1).DimensionCount -
1      intPC0 = cst.Axes(0).positions.Count - 1     intPC1 =
cst.Axes(1).positions.Count - 1      ' Columns     For y = 0 To intDC0         sXML =
sXML & "<row>"         For c = 0 To intDC1             sXML = sXML &
"<column></column>"         Next         For x = 0 To intPC0             sXML = sXML
```

```
& "<column>" & CleanString(cst.Axes(0).             positions(x).Members(y).Caption)
& "</column>"          Next         sXML = sXML & "</row>"      Next          ' Rows
For y = 0 To intDC1         For x = 0 To intPC1             sXML = sXML & "<row>"
For r = 0 To intDC1                 sXML = sXML & "<column>" &
CleanString(cst.Axes(1).                  positions(x).Members(r).Caption) &
"</column>"             Next             For z = 0 To intPC0                 sXML =
sXML & "<column>" & cst(z, x).Value & "</column>"             Next             sXML =
sXML & "</row>"         Next     Next
```

I have used a function to clean each string, in case you might wish to open the resultant XML file within an XML editor. Certain characters are best removed in XML files, which the function automates.

```
Function CleanString(str)    Dim charList  Dim charArray       Dim i      'characters
to remove from grid element name     'add to this list as necessary      charList =
"('000);(000's);(%);($); ;$;&;(;);,;.;"";';-;+;:;/;\;@;      >;<;=;" & Chr(13) & ";"
& Chr(10)     charArray = Split(charList, ";")      For i = 0 To UBound(charArray)
str = Replace(str, charArray(i), "")     Next      'replacing special symbols/common
abbreviations     str = Replace(str, "%", "Pct")     str = Replace(str, "#", "Num")
CleanString = str  End Function
```

Hopefully, this article gave you an overview of several technologies that might be new to you. They are definitely worth checking out, if you haven't done so already.

Top 3 Tips for Improving Performance in your Crystal Xcelsius Dashboards

By Michael Alexander, author of "Crystal Xcelsius for Dummies"

"My dashboard is slow to load"

"My animations are choppy"

"There is a delay in interactivity in my dashboard"

If you any of these comments sound familiar to you, then you need to read this article. Below, we will discuss three simple things you can do to immediately improve the performance of your Crystal Xcelsius dashboards. These simple tips revolve around the effective use of what I like to call the three pillars of performance: rows, formulas and components. Each of these pillars not only plays an integral part in the efficiency of your dashboard construction, but also in the way your dashboard behaves.

The Three Pillars of Performance: Rows, Formulas, and Components

It's a fact of life that many of us deal with processes and models that are inherently complicated. When we see the benefits that visualizing our data with Crystal Xcelsius can provide, we jump right in and try to convert our data into Crystal Xcelsius dashboards. In many cases, you are able to construct a dashboard based on your existing Excel spreadsheet with minor formatting of the data. Although this method works in most scenarios, some users run into a situation where the spreadsheet or database they are using is an overly complicated one that causes performance issues when connected to a Crystal Xcelsius dashboard. In fact some of the most common causes of slow and unpredictable performance can be resolve by tweaking a few things in your Excel model.

Tip #1: Keep a check on the number of Rows you are using

You would be surprised at the number of users I encounter that use Crystal Xcelsius as a presentation layer to their database. And by that I don't mean they use Crystal Xcelsius with Microsoft Access or SQL Server. I mean that they literally extract a huge chunk of raw data, jam that data into Excel, then try to build a model around it.

Remember that Crystal Xcelsius is a vehicle for presenting analyses, not raw data. The data that works best with Crystal Xcelsius is data that has already been aggregated and summarized into useful views that can be navigated with selectors, map components or any other component.

Excel models that contain tens of thousands of rows will cause your dashboard to load slowly and possibly even slow down in runtime. So the question is: how many rows are too many? The answer to that is a bit nebulous, as it depends on what you are doing with those rows. I've seen models with 23,000 rows of data run just fine; but here's the catch. Those models had two things going for them: they contained only a handful of columns, and there were no complex VLOOKUP or concatenation formulas tied to each row.

The bottom line is that you should always perform as much data aggregation and massaging you can before you build that data into a model.

Tip #2: Limit the use of Formulas

First off, let me say that using Excel formulas to enhance your dashboards is a good thing. There is no better way to add functionality to your visualizations than to use a handful of strategically placed formulas.

That being said, there are some serious performance issues that come with formulas. Excel itself has been optimized to run the formulas within a spreadsheet with little to no performance drag. However, you have to remember that when you use formulas in your Crystal Xcelsius dashboards, it is not Excel, but rather Flash, that is processing the commands. Although Flash engine can run most Excel formulas just fine, it does not do it in a particularly efficient way. In addition, Flash can not process all kinds of formulas. In particular, Flash seems to have trouble processing some nested formulas (formulas that use the results of other formulas as variables)

The net effect is that too many formulas in your Excel model (especially complex or nested formulas) will severely slow down your dashboard. But this article is not about problems, it's about solutions. So here are my tips for avoiding performance issues due to formulas.

Use hard-coded values where possible: Try to limit the number of formulas down to the ones that are necessary to give your dashboard its utility. Clean up any stray formulas that are not essential to dashboard and convert any one-time formulas (i.e. formulas create to get a value) to hard values.

Note: To convert a formula to a hard value, simply copy the cell that contains the formula. Once the cell is copied, right-click on the cell and select Paste Special, and then select Values.

Stay away from the CONCATENATE formula: Many people will use the CONCATENATE formula to string together values for use in VLOOKUPs. These formulas must remain dynamic so they can not be converted to hard values. Unfortunately the CONCATENATE formula itself seems to be especially problematic for Flash to process, often slowing the model down when too many are present in the model. The solution: replace the CONCATENATE formula with a simple reference formula that uses the ampersand. For example: Instead =CONCATENATE(A1,B1,C1), use =A1&B1&C1. The alternative has the same effect without the performance drag.

Avoid the use of nested formulas: Although many of your nested formulas will work, nested formulas generally slow down performance and, in extreme cases, can cause some unpredictable behavior in runtime. For example, have you ever had a chart or a gauge disappear when you run your dashboard? Chances are, the problem you were facing was that the formula feeding the component was a nested formula that got its value from other nested formulas. Ideally, you want to set up your model to avoid the need to use formulas as variables of other formulas.

Tip #3: When it comes to components, less is more

It's natural to try to keep your visualizations within one dashboard. In our minds, keeping track of one dashboard is much simpler than using separating our analyses into multiple modules. However, some dashboards are so complex that they require many components to achieve the required look and feel. Many users even use dynamic visibility to be able to pack as many components as possible on one dashboard.

As you can guess, this is problematic for two reasons. From a practical standpoint, having too many components on one canvas can make managing and maintaining your dashboard a nightmare. From a performance standpoint, every component you add to your visualization is one more component that has to

load when you activate your dashboard. That is to say, the more components you have, the longer the dashboard will take to load.

So what's the solution? I recommend you split your dashboard up into multiple models when possible, each of which contains a specific function. Then you can then bring all the Flash files into one "master" dashboard using the Image component. In addition to JPG files, you can also import SWF flash files for use in your dashboards. This allows you to nest existing Crystal Xcelsius SWF files into your visual model.

Here's how it works.

1. Create a dashboard that does something, then export it out as a SWF file.
2. Start another dashboard and drag an Image component onto the canvas.
3. In the properties of the Image component, use the File Name Property to select the SWF file you just loaded.
4. Run the dashboard in Preview mode to see the embedded dashboard in action.

The idea is to split your gigantic dashboard up into smaller dashboards that perform separate tasks. Save each dashboard as a SWF file, then embed each one into an empty dashboard using the Image component. From there, you can set dynamic visibility on each image component to ensure only one is loaded at a time. Crystal Xcelsius loads only one embedded visualization at any given time, which means your dashboard will load faster and perform better.

Note: When you import an SWF file using the Image component, the SWF file will become attached to the visual model, but only as an external file with references. When you export the final dashboard, Crystal Xcelsius will output the nested SWF file into a directory with the final dashboard. These files must be kept with the final dashboard in order for the dashboard to work properly. If you move the dashboard, you must move all accompanying files with it.

Using these three simple tips you will be able to immediately improve the performance of your Crystal Xcelsius dashboards. If you are a power user that needs additional help, take heart—more solutions do exist. To learn more, I encourage you to contact Crystal Xcelsius technical support or visit www.datapigtechnologies.com to find out what else you can do to eliminate performance drag on your models.

Conditional Formatting in Charts without Alerts

By Michael Alexander, author of "Crystal Xcelsius for Dummies"

Using Alerts in a chart is a great way to incorporate conditional formatting, allowing your users to quickly measure performance—good or bad. In Excel, there is no such function as alerts in charts—or conditional formatting for that matter. Before Crystal Xcelsius came along, any conditional formatting had to be done with some clever charting tricks. I've found that because Crystal Xcelsius supports many of the existing functions in Excel, several of my old charting tricks work quite nicely in my dashboards.

Note: The source files for this article can be found here.

One such example is my old trick to highlight the minimum and maximum values in a chart (seen here in Figure 1).

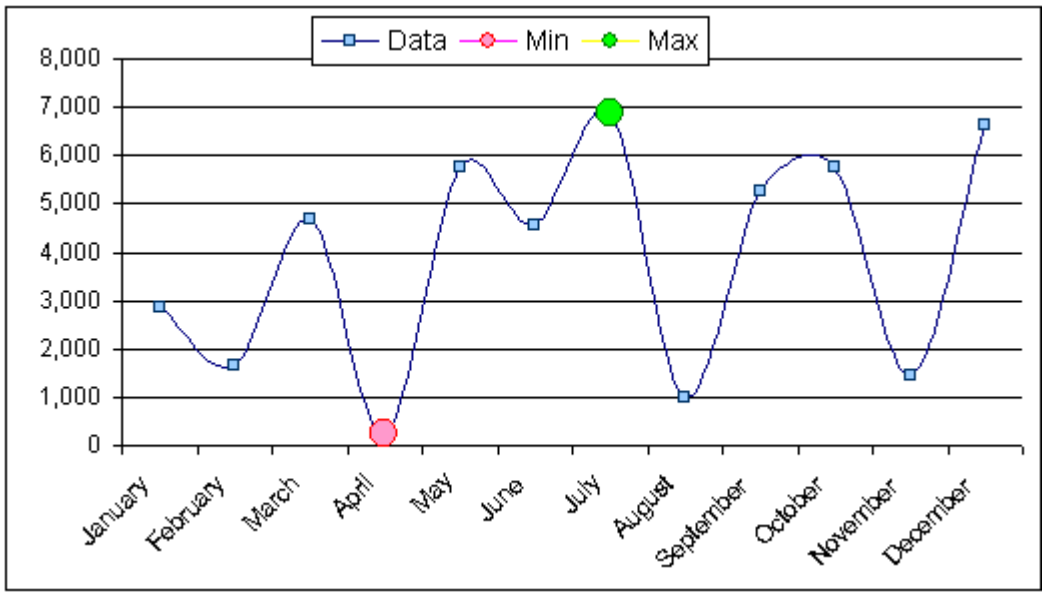


Figure 1: An old trick in Excel, allows you to format the minimum and maximum values in a chart

I tried this trick in Crystal Xcelsius and I found that I could apply some pretty cool conditional formatting to my chart without the use of alerts. Notice in Figure 2, my chart not only highlights the min and max values, but I've added an interesting effect where these values are represented with a star.

Figure 2: The same trick applied to Crystal Xcelsius

To get this effect, I started off with a table of data and I added two columns: a Min column and a Max column.

C	D	E	F
	Data	Min	Max
January	6421		
February	1200		
March	5627		
April	4540		
May	4674		
June	5052		
July	407		
August	2564		
September	4775		
October	2524		
November	5961		
December	5604		

Figure 3: Start with your data table and add min and max columns

Next, I enter a formula in the first row of the Min column that checks to see if the data on that row is the minimum value for the entire dataset. If it is, I use the value; if it's not, I simply fill the cell with a null string (represented by two quotes). So in the example demonstrated here in Figure 4, I check to see if cell D3 is the minimum value of the entire range of data (D3:D14). If it is, I use the value in D3; otherwise I fill the cell with a null string ("").

=IF(D3=MIN(D3:D14),D3,"")

C	D	E	F
	Data	Min	Max
January	6421		
February	1200		
March	5627		
April	4540		
May	4674		
June	5052		
July	407		
August	2564		
September	4775		
October	2524		
November	5961		
December	5604		

Figure 4: Write the formula that finds the minimum value

After the formula is written, I copy it down to all rows in my dataset. Notice in Figure 5 the formula contains dollar signs in the range (D$3:D$14). This tells Excel that the range is an absolute reference and it should not be adjusted when the formula is copied down.

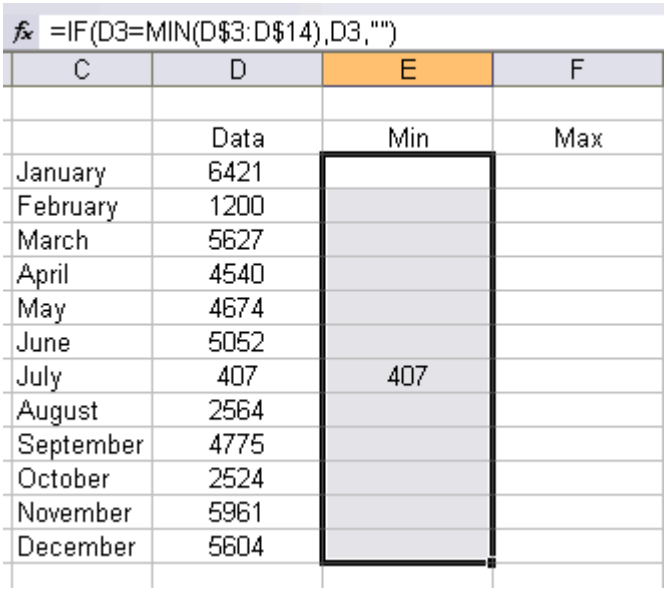


Figure 5: Copy the formula down

As you can see in Figure 6, I did the same thing for the Max column, only I used the MAX function.

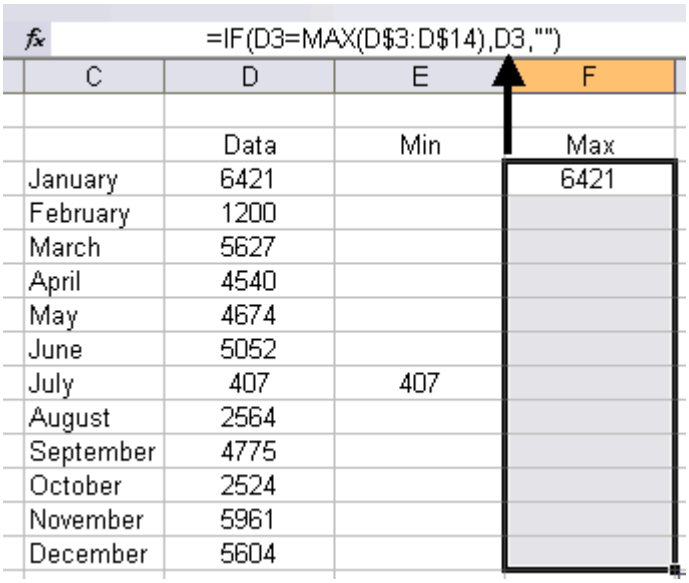


Figure 6: Fill the Max column in the same way

In the end, we have a table that contains 3 series: a Data series, a Min series, and a Max series. Both the min and max series should have only one value in it. This entire table will be the data source for the chart.

C	D	E	F
	Data	Min	Max
January	6421		6421
February	1200		
March	5627		
April	4540		
May	4674		
June	5052		
July	407	407	
August	2564		
September	4775		
October	2524		
November	5961		
December	5604		
	↑	↑	↑
	Series 1	Series 2	Series 3

Figure 7: The final result is a table with three data series

In Crystal Xcelsius, we can use this entire table to plot a line chart. Once the chart is created, simply change the series formatting to suit your needs by configuring the Lines & Markers property of the chart.

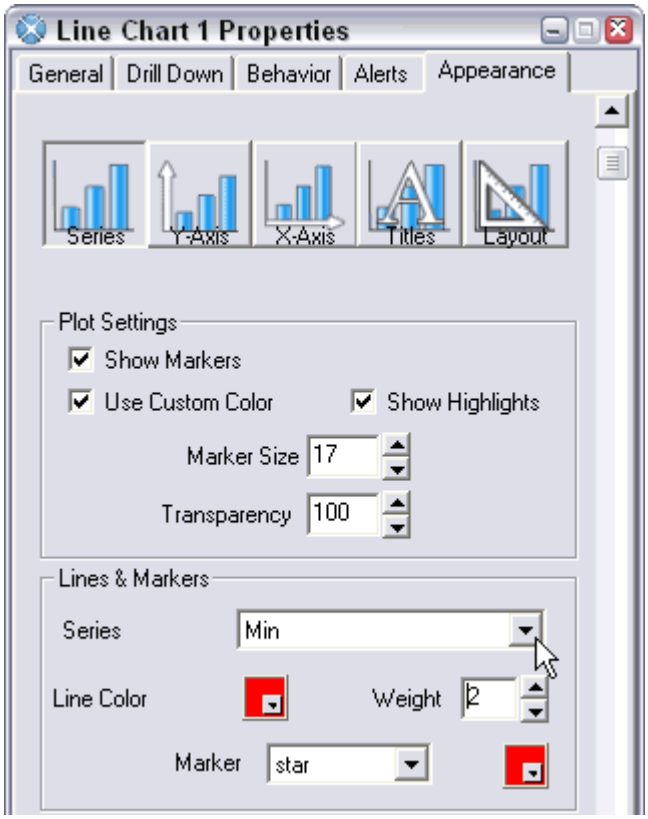


Figure 8: Configure the Lines & Markers property to add your own formatting

Keep in mind that this is just one of many examples of how an old Excel work-around can enhance your dashboards. The bigger point you should take away from this article is that you can teach and new dog old tricks. Because many of the functions in Excel are exposed through Crystal Xcelsius, you will find that most of your clever Excel tricks will work in your dashboards. So take a moment and think about your old Excel work-arounds and try them out in Crystal Xcelsius. You may be pleasantly surprised!

Add Interactive Flash Files to your Xcelsius Dashboards

By David McAmis, BI Evangelist, Avantis Information Systems Pty Ltd

Did you know that you can embed Flash files into your Crystal Xcelsius dashboards and data presentations? This article shows you how to add some pizzazz to your Crystal Xcelsius models using the Image Component. You'll learn how to find and integrate Flash components into your models—resulting in some exciting effects like dynamic sliders, hidden boxes, menus and more.

Note: Download the resource files which accompany this article.

If you have worked with Crystal Xcelsius much, you already know that it leverages the Adobe Flash platform to deliver dashboards and data presentations. Flash provides a very visual, feature-rich environment, and Crystal Xcelsius is a great example of what the Flash platform was designed to do—deliver data in an engaging and visually compelling way. But did you know that you can also incorporate other Flash elements into Crystal Xcelsius?

Here's how it works: One of the components you can add to your Crystal Xcelsius dashboards is the "Image Component" If you select View > Components, the image component is available under the Art & Backgrounds folder. You can simply drag and drop it on to your visualization.

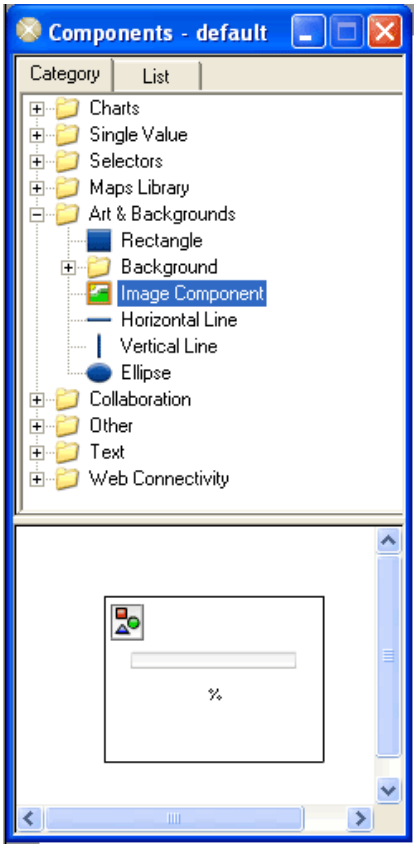


Figure 1: Selecting the Image Component from the Components Menu

There are two different types of graphic formats you can add using this component—the first is JGP and the second is SWF. SWF files are the native file format used by Flash and can be created by a number of different applications, including Adobe's Flash Professional 8. Creating Flash files from scratch is a daunting task and not really recommended for mere mortals—but the good news is that you don't need to know how to create Flash files in order to use them in your Crystal Xcelsius model.

Crystal Xcelsius includes a number of ready-to-use SWF components that you can use immediately while creating your dashboard or presentation. These SWF components include animated backgrounds, icons, sliding panels, etc. And there are a number of applications that generate SWF files as well, including applications to record demos, survey users, etc.

In the files accompanying this tutorial, there is a component named "Display Cover.SWF" that you may have seen used before in Crystal Xcelsius samples. This SWF file displays a tab which the user can then click on to reveal the components displayed underneath, as shown in Figure 1 below.

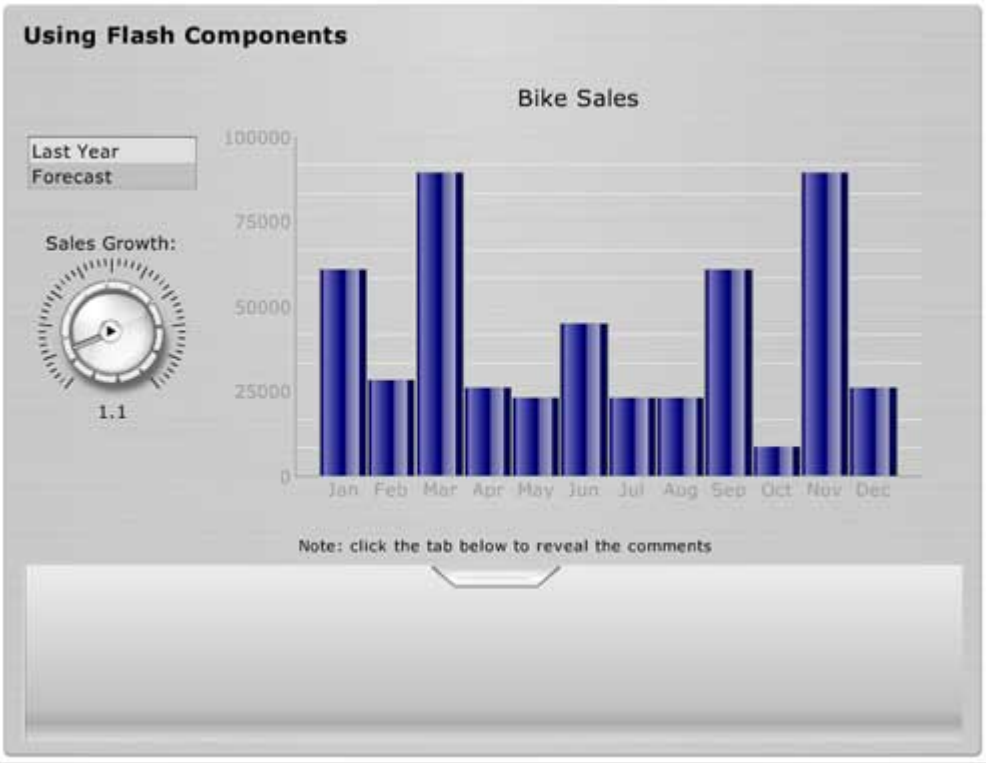


Figure 2: The tabbed control at the bottom of the page is an example of a SWF image added to a dashboard

To embed this SWF file using the Image Component, follow these steps:

1. Create a new Crystal Xcelsius project and import a spreadsheet model
2. Select View > Components and locate the Image Component under the "Arts and Backgrounds" folder
3. Drag the Image Component on to your canvas
4. Right click on the Image Component and select Properties
5. Click the Import button to browse and select your SWF file (in this case, DisplayCover.swf)
6. On the properties page, change the transparency to 100% so you can see what is behind the panel

That's all there is to it! Using SWF files is quick and easy—the trick is finding high quality files you will want to use in your dashboard or presentation. The best place to start is to do a Google search for "SWF Clip Art" or "SWF Components". There are a number of them out there—one of my favorites is Quick Poll from Flash Relief (www.flashrelief.com). Quick Poll can be used to add a poll to a Crystal Xcelsius dashboards and then immediately show the results. For example, you could have a user pick their choice for where to hold the Christmas party and then immediately show them the results. See another example here.

Another neat way to add SWF files to Crystal Xcelsius is through Adobe Captivate (www.adobe.com/products/captivate/) which can be used to create presentations, walk-thru's, training movies, etc. all of which can be exported to SWF. This is an excellent way to add PowerPoint slides, narration, etc. to your dashboards, either as a main component of the model, or as part of a help system to guide users.

Using Flash graphics and audio is a quick and easy way to add a totally new dimension to your Crystal Xcelsius dashboards and data presentations. Give it a try—you will be pleasantly surprised with the results!

Creating Dynamic Charts with Formulas in Xcelsius

By Michael Alexander, author of "Crystal Xcelsius for Dummies"

Dynamic Visibility is a feature within Crystal Xcelsius that lets the dashboard designer hide or show certain components based on a certain value (i.e. "1" or "0") found in a particular cell. When used in conjunction with selectors—such as the Label Based Menu or Toggle Button—that can insert numbers into a cell, the designer can create a multi-layered dashboard that lets the user quickly open or close various panels within the overall model. (For more about dynamic visibility, click here)

In the article, we'll be focusing on a clever workaround that lets you build dynamic dashboards, without using dynamic visibility. Take a look at the dashboard shown here in Figure 1. While it may seem as though this dashboard is using dynamic visibility to cycle between three separate charts, we are actually using only one. The chart is dynamically fed different data sources with the use of a few simple formulas. In this article you will discover how a few clever formulas can help you simplify your dashboard design, which can help improve its overall performance.

Why avoid multiple charts and dynamic visibility?

Let's take a moment to talk about why you would want to avoid multiple charts and dynamic visibility. The datasets used to create this dashboard are shown here in Figure 2. As you can see, each the dataset is comprised of different information, yet their formats are very similar.

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7			Q1	Q2	Q3	Q4
8		2005 Income	$399,354	$573,662	$244,661	$790,906
9		2004 Income	$219,967	$495,072	$212,749	$687,744
10		2003 Income	$159,832	$289,825	$181,961	$456,016
11						
12		2005 Expense	$219,967	$495,072	$212,749	$687,744
13		2004 Expense	$219,468	$310,048	$307,124	$283,920
14		2003 Expense	$71,744	$607	$119,251	$154,487
15						
16		2005 Net	$179,387	$78,590	$31,912	$103,162
17		2004 Net	$498	$185,024	-$94,375	$403,824
18		2003 Net	$88,088	$289,218	$62,710	$301,529

Figure 2: Our datasets are different, but have similar formats

Now, you could create three separate charts and show them all on your dashboard at the same time—however, this method has its flaws. The first drawback to this option is that you will take up valuable real estate by showing all three charts. Also, you may inundate your clients with too much information at one time. Thirdly, any changes made to the formatting in your Excel spreadsheet would have to be performed on all three charts, making time spent on maintenance an issue.

An alternative is to create three separate charts and use dynamic visibility to manage the way your clients see the charts. This takes care of the real estate problem by allowing you to stack the charts on top of each other, showing only one chart at a time. However, you will still need to maintain, and keep track of multiple charts. In addition, you will need to document and manage the dynamic visibility of each chart to ensure that works properly.

That being said, it's important to point out that dynamic visibility is a very valuable tool when you want to manage multiple charts that have very different structures and dimensions. However, when there is a need to

plot datasets that are similar in structure, as in Figure 2, then you can avoid the maintenance and management of multiple charts with dynamic visibility.

Creating a Dynamic Chart with Formulas

Note: Download the source files for this article.

The first thing we will do is start a new dashboard and add a Label Based Menu. After adding the menu, we will change the labels to reflect the different datasets we will toggle through in our dashboards. Figure 3 demonstrates how. As you can see, in this scenario, we want our users to toggle between Income, Expense, and Net.

Notice that the Insert Option property is set to Position. This is very important. This option tells the menu to output the position number of the label that is selected. For example, if "Income" is selected, then the number 1 will be output because "Income" is the first label. If "Net" is selected, then the number 3 will output because "Net" is the third label. The output number that sent from our Label Based Menu will determine which dataset is used to feed our chart.

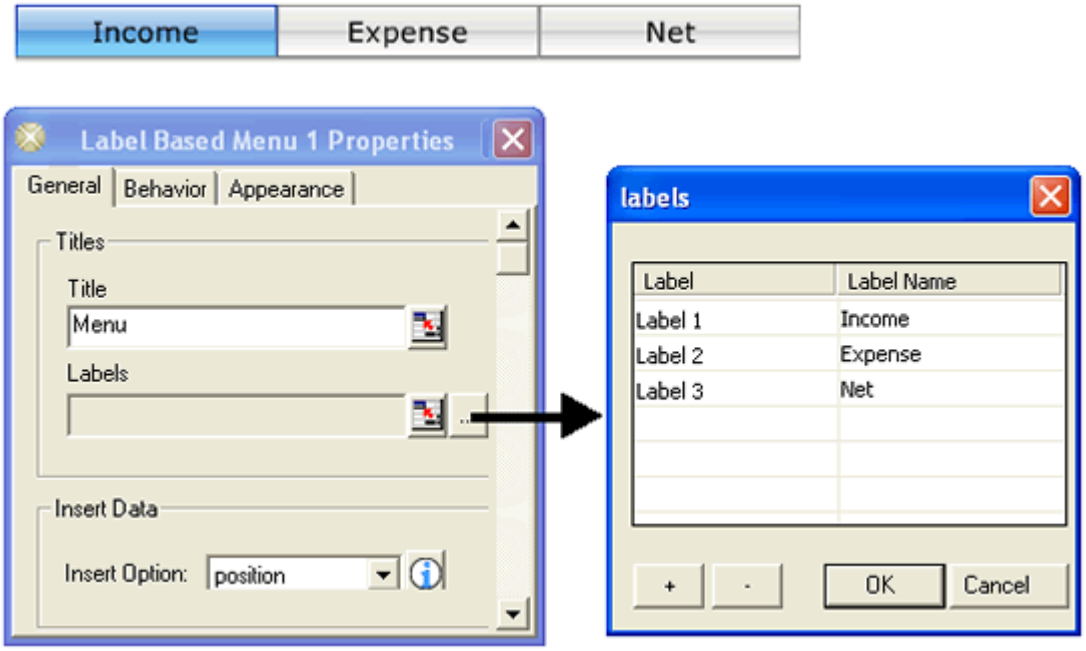


Figure 3 caption: Add a Label Based Menu and set the labels

We then set the Insert Into property to feed the menu's output into Cell B1 as shown here in Figure 4. This ensures that each time a user toggles through the menu, cell B1 will change to reflect the label that was selected.

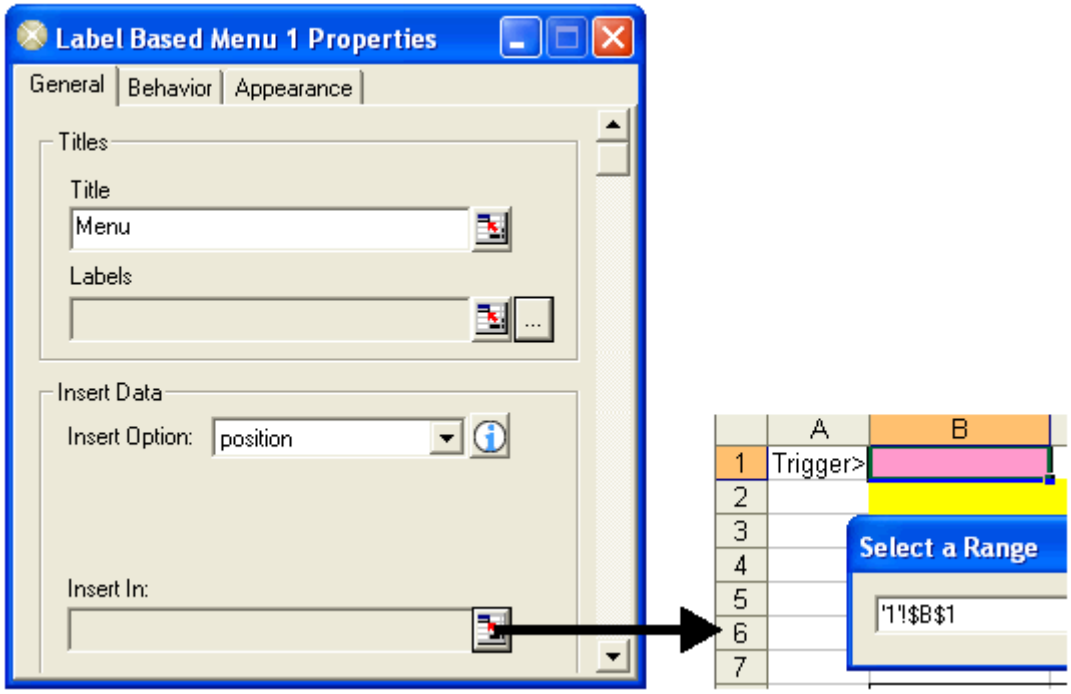


Figure 4 caption: Edit the Insert Into property to feed cell B1

Next, we turn our attention to the Excel model. The idea is to create a dynamic dataset, shown here in Figure 5 in yellow. What make this dataset dynamic is that we will not fill the dataset with hard-coded values. Instead, we will use a set of IF formulas to fill the dataset.

B3 =IF($B$1=1,B8,IF($B$1=2,B12,B16))

	A	B	C	D	E	F
1	Trigger>	1				
2			Q1	Q2	Q3	Q4
3						
4						
5						
6						
7			Q1	Q2	Q3	Q4
8		2005 Income	$399,354	$573,662	$244,661	$790,906
9		2004 Income	$219,967	$495,072	$212,749	$687,744
10		2003 Income	$159,832	$289,825	$181,961	$456,016
11						
12		2005 Expense	$219,967	$495,072	$212,749	$687,744
13		2004 Expense	$219,468	$310,048	$307,124	$283,920
14		2003 Expense	$71,744	$607	$119,251	$154,487
15						
16		2005 Net	$179,387	$78,590	$31,912	$103,162
17		2004 Net	$498	$185,024	-$94,375	$403,824
18		2003 Net	$88,088	$289,218	$62,710	$301,529

Figure 5 caption: Designate the location cells for your dynamic dataset. Highlighting the cells for quick identification is a recommended best practice

In the first cell of our dataset we will enter the following formula:

=IF($B$1=1,B8,IF($B$1=2,B12,B16))

This formula tells Excel to check the value of cell B1 (the cell where the Label Based Menu will send the output number). If the value of cell B1 is 1, which represents the value of the "Income" label on our menu, then we grab the value in the "Income" dataset (cell B8). If the value of cell B1 is 2, which represents the value of the "Expense" label on our menu, then we grab the value in the "Expense" dataset (cell B12). If the value of cell B1 is not 1 or 2, then we grab the value in cell B16.

Note: Notice that in the formula, we are using absolute references with cell B1. This means that we are prefixing the column and row reference with $ signs ($B$1). This will ensure that the cell references in our formulas don't shift as we copy them down and across.

As you can see in Figure 6, we simply copy the formula across and down to fill the dataset. To test that everything is working fine, we can change the value of cell B1 to see that the values in our dynamic dataset change.

	A	B	C	D	E	F
1	Trigger>	1				
2			Q1	Q2	Q3	Q4
3		2005 Income	$399,354	$573,662	$244,661	$790,906
4		2004 Income	$219,967	$495,072	$212,749	$687,744
5		2003 Income	$159,832	$289,825	$181,961	$456,016
6						
7			Q1	Q2	Q3	Q4
8		2005 Income	$399,354	$573,662	$244,661	$790,906
9		2004 Income	$219,967	$495,072	$212,749	$687,744
10		2003 Income	$159,832	$289,825	$181,961	$456,016
11						
12		2005 Expense	$219,967	$495,072	$212,749	$687,744
13		2004 Expense	$219,468	$310,048	$307,124	$283,920
14		2003 Expense	$71,744	$607	$119,251	$154,487
15						
16		2005 Net	$179,387	$78,590	$31,912	$103,162
17		2004 Net	$498	$185,024	-$94,375	$403,824
18		2003 Net	$88,088	$289,218	$62,710	$301,529

Figure 6 caption: Copy the formula across and down to fill the dataset

After saving our changes in the Excel model, we can add a chart to our dashboard and reference the dynamic dataset. Figure 7 demonstrates how.

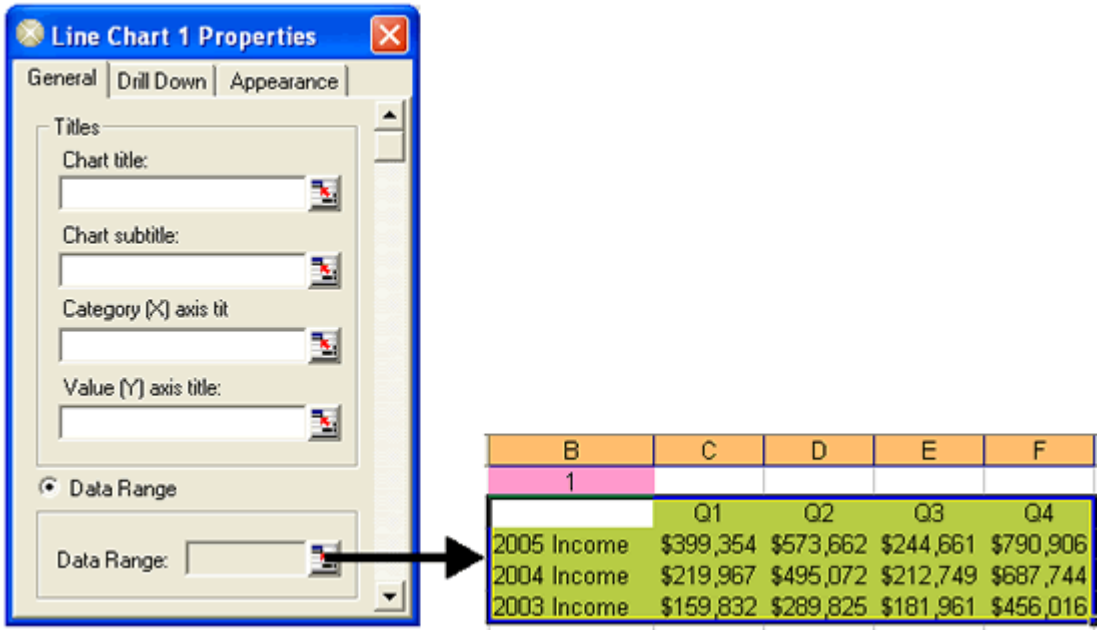


Figure 7 caption: Add a chart to your dashboard and reference your dynamic dataset

The reward for your efforts will be an interactive dashboard consisting of one chart and one label-based menu. As you click on a label in your menu, the chart changes. Again, the major benefits you get from this type of setup is that any formatting changes can be made to one chart, you have no dynamic visibility to manage, and can easy add another dataset by expanding your menu and editing your formula.

It's important to keep in mind that this type of setup works well when you have multiple datasets that are similar enough to share the same chart. Datasets that vary in structure and dimension typically don't lend themselves to this technique. However, as you think about the dashboards you have created, you may discover that a there are some that can be enhanced using this type of setup.

Creating a Connected Model Using the Web Services Component in Xcelsius

By David Harper, Principal, Investor Alternatives, LLC

Introduction

One of the key features of a dashboard or data presentation built with Crystal Xcelsius Workgroup is its ability to refresh itself with new metrics, either at preset intervals, or on-demand with the push of a button. This feature allows us to create truly interactive models, ones that are powered by source data beyond our original Excel spreadsheet.

Explaining how to create a full-scale connected dashboard goes beyond the scope of this article, but we can demonstrate how the concept works using a simple example that's sure to appeal to frequent flyers: An interactive map that plots the location for any airport inside the United States.

Tip: Download the source files for this article.

Any Crystal Xcelsius Workgroup model, including our example here, can connect to external data sources via the use of web services.

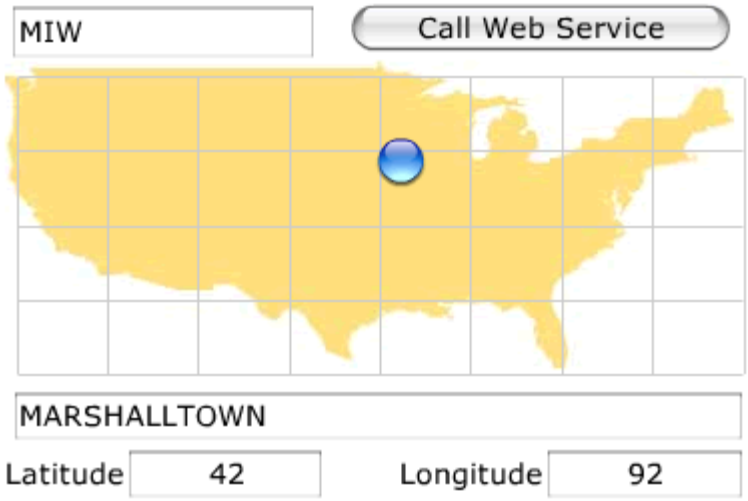


Figure 1

You can invite trouble by trying to define the term web services, but let's do it anyway: *a web service is an application that can send and receive messages*. Generally, the messages are sent in eXtensible Markup Language (XML), an important standard that allows different machines to communicate.

This may sound complicated, but never fear: Crystal Xcelsius buries most of the complexity. In our example here, the user needs only to input the airport code; e.g., SFO for San Francisco, MIA for Miami International. We only need to ask the **Web Services Connector Component** to make a request from a web service. As illustrated below, the component will send a request (i.e., the input) message and it will receive back a reply (i.e., the output). Both messages use the XML format.

	A	B	C	D	E
1					
2					
3			X Axis	Y Axis	Size
4	Series 4	Red Light>>	50	75	1
5					
6	Series 3	Yellow Light>>	50	50	1
7					
8	Series 2	Green Light >>	50	25	1
9					
10	Series 1	Dummy Lights >>	50	75	1
11		Dummy Lights >>	50	50	1
12		Dummy Lights >>	50	25	1

Figure 2

An amazing fact is that we need to dedicate just two cells in our Excel source file (airport.xls) to handle this entire exchange of information. In cell F1, we will allow for the input of the airport code. We will designate cell F2, left blank at model construction, as the cell that receives the "element" from our incoming reply message.

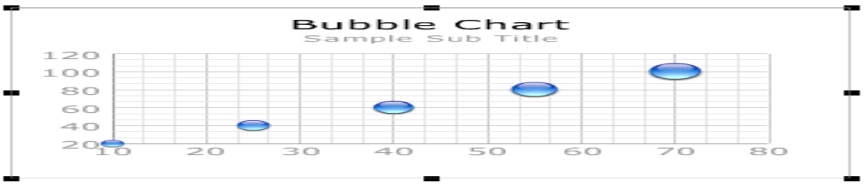


Figure 3

Binding the Data

The critical step is to "bind" the web service elements to cells in the spreadsheet, as the elements themselves are just parts of the send/reply message that contain data.

Below is a snapshot of the Crystal Xcelsius Web Services component. The first thing required is a URL address for the service. That is what a public web service looks like, a simple URL. The service will offer one or more methods (functions). In our case, we want the getAirportInformationByAirportCode function. There are many web services, some of free, but the best ones require a subscription. For this example, I used a free service at www.webservicex.net that accepts an airport code (i.e., the input message) and returns a large XML file filled with information about that airport (i.e., the reply). The location of the specific service is http://www.webservicex.net/airport.asmx?WSDL:

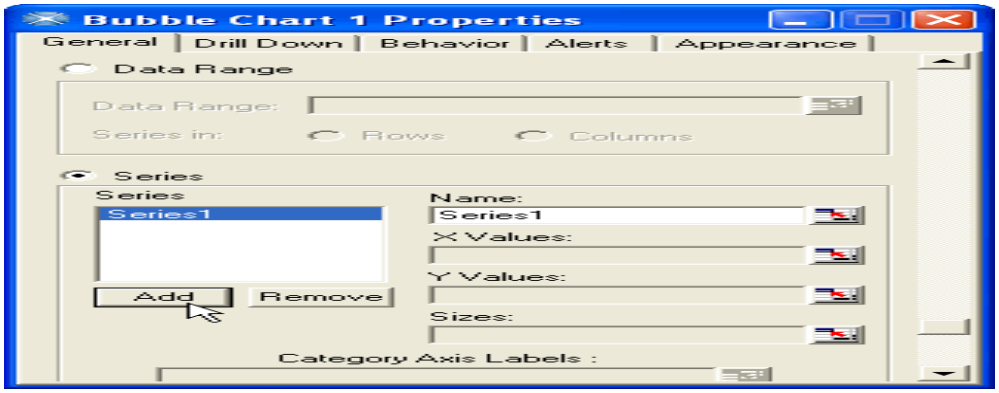


Figure 4

Once we hit "submit," Crystal Xcelsius populates the dialog with both input/output messages and their respective elements. For the purposes of this example, I chose a basic service that has one element each. But, if you have multiple elements, you can map each of them to different cells in your spreadsheet.

We map the sole Input Value element (airportCode) to cell $F$1 and the sole Return Value element (getAirportInformationByAirportCodeResult) to cell $F$2. Again, this is referred to as *binding*: when you map

data elements to components or sub-components in your application. In this case, the subcomponent is the spreadsheet cell.

Binding the Input Value element to cell $F$1

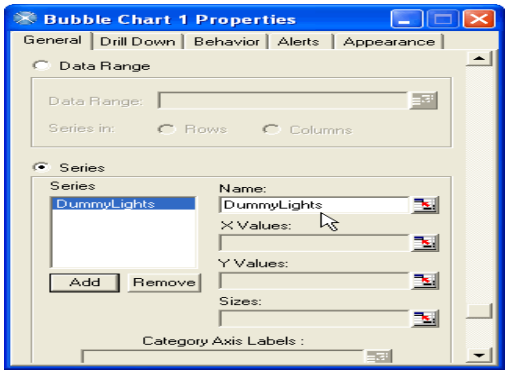


Figure 5

Binding the Return Value element to cell $F$2

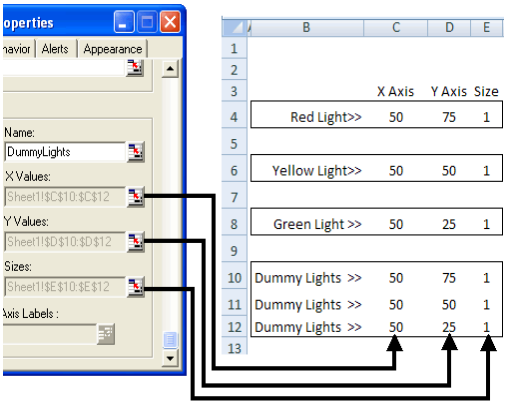


Figure 6

Returning the XML file

The only glitch with this example is that the returned element is an XML file. Often, you will have several elements in the return and each element is its own value (e.g., stock price, revenue). In that case, you simply bind as above and your spreadsheet cell will be ready for use.

For our example, the returned XML file looks as below. While at first glance, XML files may appear overly-complex, please do not be intimidated. There are two basic parts to an XML file, the tags and the values within the tags:

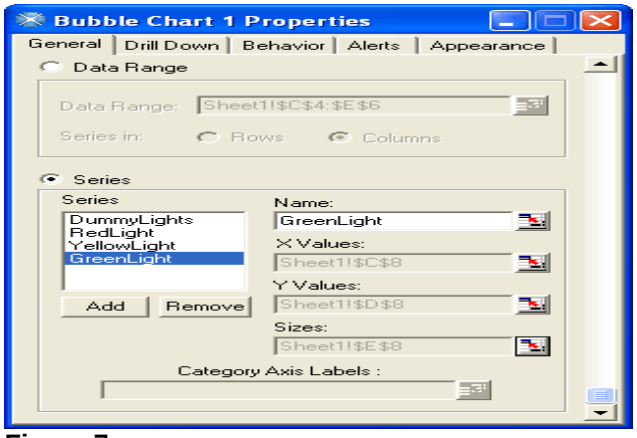


Figure 7

Because the XML file follows a standardized structure, I can predict where our reply tags will be. In this case, I only want the latitude and longitude numbers. By following the XML structure, we know that the latitude value will be enclosed by <LatitudeDegree> and its corresponding closing tag. Longitude is similarly enclosed by <LongitudeDegree>. To parse out these numbers, then, I used the FIND() function to locate their position, followed by the MID() function to extract the numbers themselves.

Once I have these numbers parsed/extracted, I simply overlay a bubble chart above the map. The spreadsheet is quite small because the Crystal Xcelsius component does the heavy lifting.

Conclusion

Integrating web services inside a Crystal Xcelsius model takes some practice to master—however, once you understand the technology, you will be able to create truly dynamic dashboards that combine the power of real-time data with the presentation capabilities of Crystal Xcelsius.

Using Crystal Xcelsius with Microsoft SQL Server Reporting Services

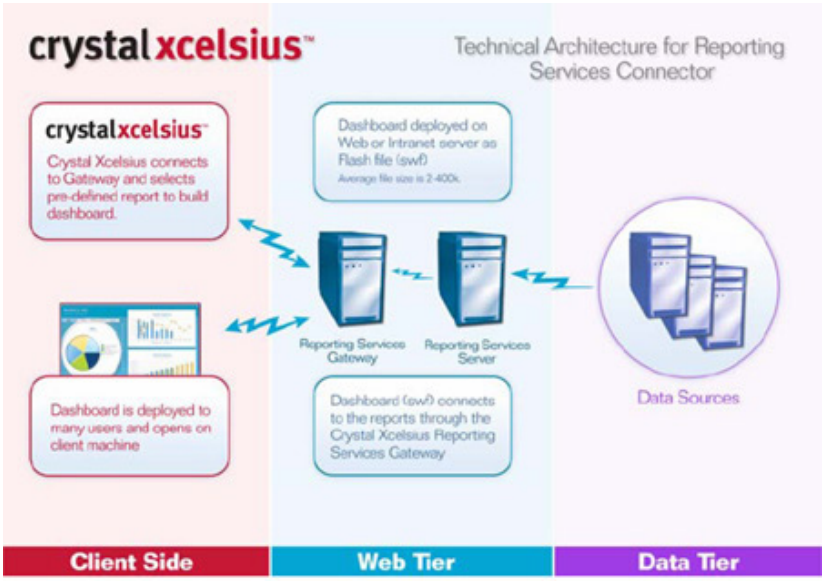
By David McAmis—BI Evangelist, Avantis Information Systems Pty Ltd.

The following is an excerpt from Using Crystal Xcelsius with Microsoft SQL Server Reporting Services. Download the entire white paper here.

Crystal Xcelsius with Connectivity to Reporting Services

With point-and-click connectivity to Microsoft SQL Server Reporting Services, Crystal Xcelsius provides a new and powerful solution for Microsoft SQL Server 2000 and SQL Server 2005 Reporting Services customers.

Installing the Crystal Xcelsius component on your Reporting Services server lets you access your reports from inside the Crystal Xcelsius interface and connect report data directly to the dashboard, replacing ranges within the Excel data model you used to build the dashboard. This easy access and connectivity ensures your business dashboards are always live with up-to-the-minute information.



Features

Crystal Xcelsius with Reporting Services connectivity gives you:

Point-and-Click Connectivity
Connecting to Reporting Services new and existing reports and creating dashboards helps you to make optimal use of key features such as security—all with point-and-click ease.

Tight Integration
View a list of available reports, view the report parameters and launch a report preview all from within the Crystal Xcelsius user interface.

Dynamic Data
Data is always up to date, with the option to refresh the data when the dashboard is loaded or at a set time interval. User can also refresh a dashboard at the push of a button and can be prompted for any report parameters.

Stunning Visualization
Custom graphics and animation provide you with compelling, visually appealing business dashboards—dashboards that can be used for Web sites, intranets, portals, and any environment that supports Flash.

Rapid Development and Deployment
The intuitive user interface combined with point-and-click connectivity lets you develop and deploy secure and interactive business dashboards in days versus months.

Xcelsius for Developers: A Technology Overview and Introduction

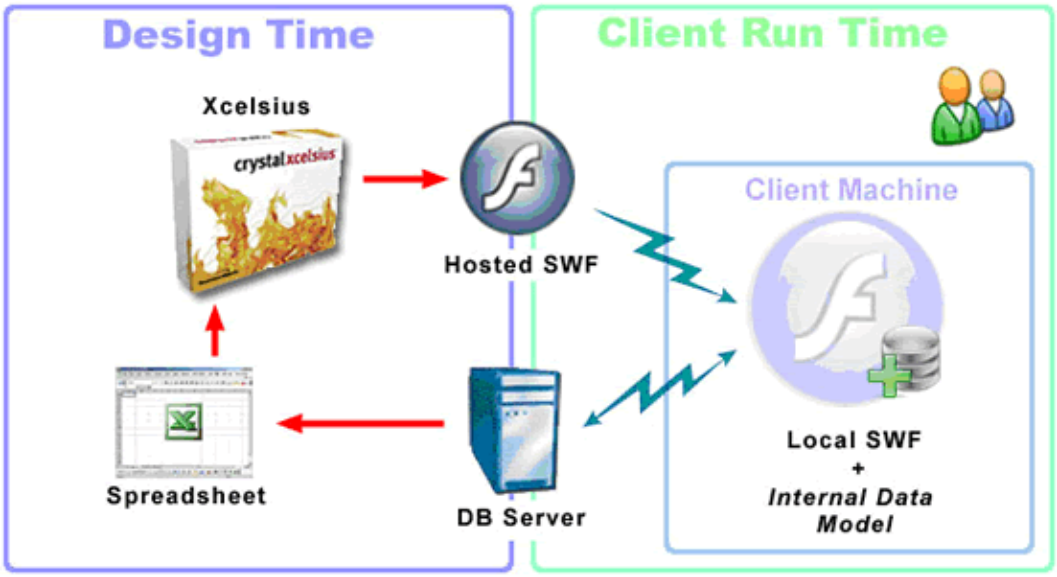
By Thomas Gonzalez, Managing Director of BrightPoint Consulting, Inc.

The following is an excerpt from Crystal Xcelsius for Developers: A Technology Overview and Introduction.

How does it work?

Crystal Xcelsius uses a unique combination of Microsoft Excel, as a data modeling tool, and Macromedia Flash Player, as the rendering engine, to compile dynamic visualizations via its proprietary Crystal Xcelsius Designer. Before you dismiss CX as a toy because it uses Excel for data modeling, let me emphasize the word DESIGNER in the previous sentence. Within Crystal Xcelsius, Excel is used solely as a data design tool to help integrate and manipulate data into a format that is easy to bind to and that is used for common charting and data visualization components. We will discuss the value of Excel as a data modeling tool a little later on, but first I want to show, from an architectural perspective, how all the technology pieces fit together.

Due to the nature of Flash and the way Crystal Xcelsius leverages it, the technology architecture and deployment model are unique and not immediately obvious to the developer more accustomed to the typical client/server architecture of most BI and reporting platforms. The diagram below shows the relationship between Excel, the Crystal Xcelsius Designer, Flash, and the client. It is important to note the different role each component plays at design time versus run time.



At design time, Excel is used to model and format data while the CX Designer is used to visualize and bind Excel-modeled data to interactive visualizations via the Flash engine. Once a design is ready to be deployed or previewed, Crystal Xcelsius then does its "magic" and compiles the data model and visualization elements into Flash byte code in the form of a .swf file. This .swf file now contains a virtual representation of the data and its relationships as defined in the Excel model, as well as all the graphic assets and animations bound to this data. The .swf file also contains information for any XML data connections that were specified at design time. These data connections can be used to replace the existing data you designed within Excel with different sets of data at run time, thus allowing multiple queries into larger data stores.

An example would be a sales forecast dashboard that compares current sales data to specific forecast targets. In Excel you could model both the current data and the target data, knowing that at run time the current sales data would be replaced by XML data from your database, while the target data would remain static. When Crystal Xcelsius compiles the file into Flash, it puts all the data from Excel into the .swf file, along with the XML mappings that will allow the .swf file to fetch the dynamic data. As a result, at run time the user still sees the target forecast data, stored statically within the .swf file, while also viewing the updated current sales figures, which are being fetched dynamically through the XML connection.

Once the Flash movie is running, you are no longer connected to the Excel file in any way; all data has been virtualized into the Flash byte code or is being fed by dynamic XML feeds. So you end up with an easy-to-use (Excel) and simple (row/column) metaphor for modeling and manipulating your data, without any of Excel's limitations. Once you begin using this paradigm, you realize how powerful it is: data can be massaged much more quickly with Excel than it can be programmatically.

Related Content

SAP BusinessObjects Xcelsius Community

Sample Xcelsius Dashboards

Xcelsius Add-on Marketplace

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.