



X°CELSIUS
ENTERPRISE

TechTips

Connecting Xcelsius Dashboards to External Data Sources using: An XML Data Button

A step-by-step guide to connecting Xcelsius Enterprise XE dashboards to company databases using an XML Data Button for enterprise-wide reporting.

Chris Bryant
Technical Consultant

Infommersion, Inc.
January 2005

Table of Contents

	<i>Page</i>
▪ Prerequisites for getting the most from this Guide	3
▪ Introduction	3
▪ Integrating Xcelsius Enterprise XE with your data source	4
▪ How Xcelsius works	4
▪ Basic architecture	4
▪ XML integration dynamic refresh	6
▪ XML data button	6
○ Using a script on the server to return XML data	6
○ Configuring Xcelsius Enterprise XE to refresh data	6
○ Getresults.aspx script definition	9
○ Testing the getresults.aspx script	11
○ Benefits	11
▪ XML “gotchas”	11

Prerequisites for getting the most from this Tech Tips Guide

This Tech Tips Guide is technically oriented, and as such the reader should have a solid understanding of the following technologies and terms:

Excel: A basic understanding of Excel – how to navigate and create formulas – is required. It is beneficial, but not required, to have an understanding of Excel 2003 XML Maps.

Flash: No knowledge of Flash programming is necessary

Xcelsius: Experience creating data presentations with Xcelsius is required, including an understanding of the work area components and how they work.

XML: A basic familiarity with XML – how it is structured and how it can be used for communicating between applications – is required.

Additionally, included in some guides is an example of an ASPX script. Therefore, knowledge of ASP .NET and how to connect to a database through OLEDB would be helpful for understanding this example.

A few Xcelsius-specific terms used throughout the guide:

- **Xcelsius Work Area** – The Xcelsius authoring environment – which includes menus, toolbars, components, the object browser, and the canvas – used to design and create dashboards and data presentations.
- **Xcelsius File** - The XLF file created during the development of Xcelsius data presentations. It is the working (and saved) development file used in the work area (authoring environment).
- **Xcelsius Data Presentation** – The output of the work area (authoring environment) for deployment to presentation/dashboard viewers. An Xcelsius data presentation is a Flash file, also known as a SWF file for its file extension (.swf). It can be exported to, and is fully operational in, PowerPoint, Outlook and the Web.

Introduction

A corporate dashboard is critical for monitoring the daily health of an organization. It gives decision-makers concise, visual access to key performance indicators that drive the business.

With Xcelsius|Enterprise XE, the award-winning software from Infommersion, users create and deploy corporate dashboards that make it easier than ever to identify critical data relationships, probe elaborate “what-if” scenarios, and peer into their company’s financial future.

By combining critical business data with the features of Macromedia Flash™, Xcelsius gives users the power to convert data from ordinary Excel spreadsheets and XML-compliant company databases into dynamic dashboards and data presentations for Portals, PowerPoint and the Web. This guide provides a step-by-step guide to connecting Xcelsius|Enterprise XE data presentations to company databases using Web services (dynamic Web query), for enterprise-wide reporting.

Connecting Xcelsius|Enterprise XE Dashboards to External Data Sources

Now that you've discovered the power and possibilities of building highly sophisticated and interactive data presentations with Xcelsius, let's take a closer look at how to connect a dashboard that was created with Xcelsius|Enterprise XE to external data sources – such as corporate databases – via XML.

How Xcelsius Works

Before looking at the various options for connecting an Xcelsius dashboard to an external data source, it is important to understand how Xcelsius|Enterprise XE works with Excel and why connecting to an external data source can be beneficial.

The first step to creating an Xcelsius data presentation or dashboard is to import a previously created Excel spreadsheet into the Xcelsius authoring environment (work area). This import brings in all data, formulas, formatting, mapping, and other features contained in the Excel spreadsheet, and loads it into Xcelsius' own embedded Excel spreadsheet. All the information is still contained in Excel, but now it is in an Excel spreadsheet embedded in the Xcelsius work area (XLF file). You will see this when you select a data range for use with a component. The Excel spreadsheet which was imported is opened in a temporary Excel file. Therefore, after the initial import, there is no need for the original Excel spreadsheet.

After you have created your data presentation in the Xcelsius work area, Xcelsius converts it into a Flash (swf) file. The finished presentation (Flash file) does not contain the full Excel spreadsheet as the Xcelsius work area does. It only contains the range of cells, and their dependencies, that you selected for each component. When a viewer opens the presentation (Flash file), all the necessary data is either contained within the presentation itself, or dynamically retrieved via connectivity to an external data source.

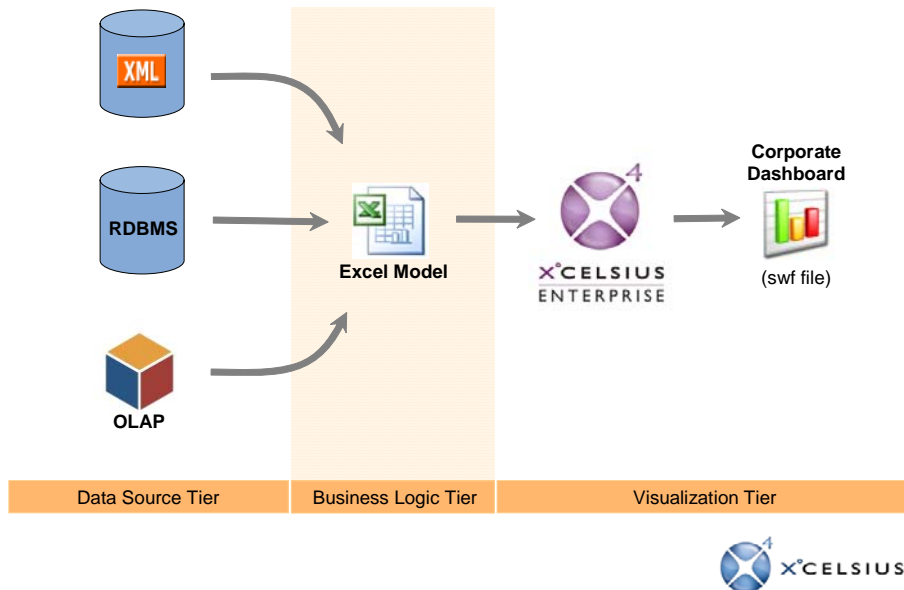
By connecting your data presentation (such as a dashboard) to an external data source, you can cause the data within the presentation to dynamically refresh – thus enabling the dashboard to be continuously connected to the latest data available.

Basic Architecture

Let's take a look at the typical architecture of how and where Xcelsius|Enterprise XE fits into a dashboard reporting environment. Following are two diagrams: design architecture and deployment architecture. The first diagram illustrates the process and technologies used to build the Xcelsius data presentation and generate a Flash file. The second diagram illustrates where the Flash file resides within the dashboard application when it's deployed.

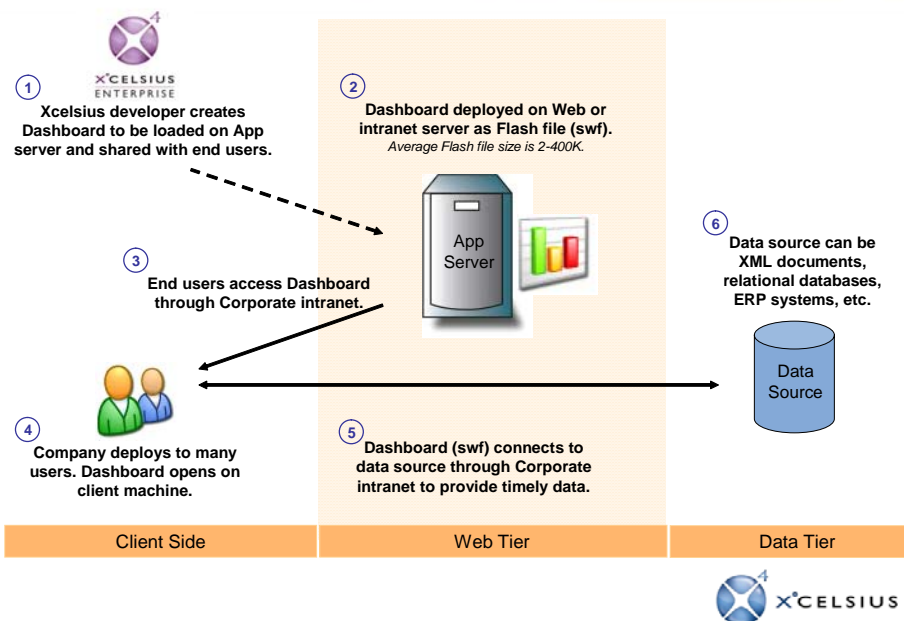
During the design phase, the data can be brought into Excel by many methods: relational pull, manual entry, importing XML, etc. Once the data is in Excel and subsequently imported into Xcelsius, there is no need for the database anymore during the design phase. All data is now self-contained within the data presentation, and the resulting Flash file can be used in static data situations.

Xcelsius Corporate Dashboard: Development



After the completed dashboard has been deployed, connectivity to the back-end data source is required to dynamically refresh the data within the Flash file. This is done independently of the original Excel spreadsheet. If there were links to the data sources created when the data was imported into Excel, those links will be passed along to the Xcelsius dashboard (Flash file), which will then be able to dynamically refresh the data through those links, this is the Excel 2003 XML Maps feature. Also, new links to different data sources can be setup within Xcelsius. These options to dynamically refresh data within a data presentation or dashboard will be discussed in the remaining sections of this document.

Xcelsius Corporate Dashboard: Deployment



XML Integration Dynamic Refresh

Now, let's take a closer look at how to connect your Xcelsius|Enterprise XE (Xcelsius) dashboard to external data sources, such as company databases.

When an Xcelsius data presentation is initially created, the data and formulas from the Excel spreadsheet are stored within the Xcelsius work area, and embedded in the presentation (Flash file). So what happens when you want the data within the Xcelsius presentation to be refreshed dynamically? Xcelsius allows you to link data presentations to an XML data source in order to retrieve updated data. This can be accomplished by three different methods: 1) XML Maps in Excel 2003, 2) Web Services (Dynamic Web Query), and 3) the XML Data button.

This Tech Tips Guide focuses on using the XML Data Button.

Note: This guide does not describe best practices for organizing your data in Excel, but is intended to help you understand various options for connecting your Xcelsius dashboard to an external data source. For more information on designing your spreadsheets for optimum use with Xcelsius, see "Xcelsius Getting-Started Tutorial: Designing Your Spreadsheets" in the online Xcelsius Learning Center.



XML Data Button

Using a Script on the Server to Return XML data

Instead of connecting to an XML file that was created from another process, it is also possible to create a Script on a Web server that runs code to connect to the database and returns the data in an XML format. We refer to this as the XML Data Button. To use it with the Web Services option you need only replace the XML file you defined with a URL pointing to your script.

This method is very similar to the Web Service (Dynamic Web Query) option. With the XML Data Button, you define a range of cells you want to be replaced by an outside data source. The XML Data Button allows the user to refresh on Load, on an Interval, or by pushing the Submit button. The other feature it provides is the ability to send a Range of cells to a script. This can be used to update the database for such applications as surveys and adding commentary to a report. For this example, we are going to show how to use a script to pull from a database and create an XML file for refreshing the data in our data presentation.

Configuring Xcelsius to Refresh data

1. For this example, you will need to have some data already saved in an Excel spreadsheet.
2. Open Xcelsius, and import your spreadsheet by clicking on the Excel icon  on the toolbar and then browsing to your desired Excel file. Click Open, then OK.
3. Click the Table component in the Components panel, and add the table to the blank slide (canvas) by dragging and dropping it anywhere on the canvas. Double Click the table to open its Properties panel and define the data to be displayed. Use the selector icon  next to the Display Data input text box to select a range of cells to show in the table. Notice the table on the canvas has updated to show the range of data you selected.

- Next you are going to add the XML Data Button to the data presentation. Select the Submit Button component from the component dialog box, and add it to the canvas.

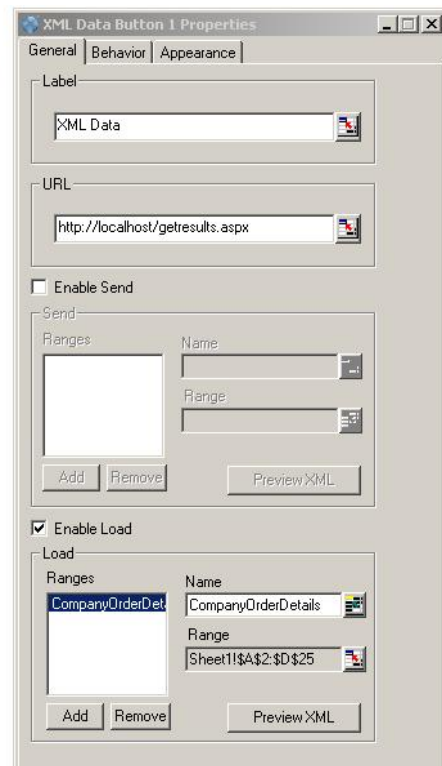
The screenshot shows a software interface with a component palette on the left and a data table on the right. The component palette is titled 'Components - default' and lists various components under 'Category' and 'List'. The 'XML Data Button' component is highlighted. Below the palette is a large 'XML Data' button. The data table on the right has the following columns: CompanyName, Unit_Price, Quantity, and Discount. The table contains 20 rows of data.

CompanyName	Unit_Price	Quantity	Discount
Laubhina Bacchus Wine Cellar	100	17	0
Old World Delicatessen	28	70	0.3
Wellington Importadora	7	20	0
LINO-Delicatesses	12.2	63	0
Reggiani Caseifici	20.1	76	0.2
Ricardo Adocicados	15.5	47	0.2
Save-a-lot Markets	2.5	16	0
Victuailles en stock	24.225	78	0.45
Great Lakes Food Market	156.4	50	0.1
Königlich Essen	90.6075	155	0.45
Cactus Comidas para llevar	26.875	27	0
Magazzini Alimentari Riuniti	20.825	40	0
Rattlesnake Canyon Grocery	38	30	0
Split Rail Beer & Ale	35.5	26	0
Trail's Head Gourmet Provisions	32.15	9	0
Drachenblut Delikatessen	32.015	32	0
Folk och få HB	12.325	21	0
LILA-Supermercado	27.125	90	0.3
Blondesd's père et fils	16	50	0
Bon app'	24.5	36	0
Island Trading	24.75	61	0
Rancho grande	172.25	7	0
Tradição Hipermercados	22.375	88	0

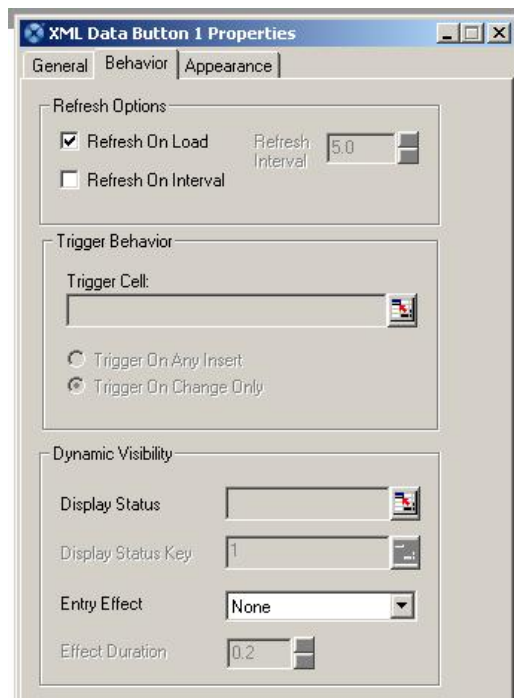
- Double Click the Submit Button to open the XML Data Button Properties window. You will notice that there are checkbox options to Enable Send and Enable Load. You can configure an XML Button to Send data, wait for a reply, and load it into the desired components. In this example, we will focus on loading data from an outside source.

First, you need to check the Enable Load box and deselect the Enable Send box. Next, you are going to define the range of data to be replaced from the script. Under Enable Load, click Add to define the range of data. The name of the range is important as it must match the variable element name in the XML file. We are using "CompanyOrderDetails" for the name of our range. Use the Excel cell selection icon on the right of the text box to select the range of cells you want to replace with data from the XML file. Click the Preview XML button to open a copy of the data in the correct XML format. Use this format when creating the XML file.

CompanyName	Unit Price	Quantity	Discount
Lauhina Bacchus Wine Cellar	100	17	0
Old World Delicatessen	28	70	0.3
Wellington Importadora	7	20	0
LINO-Delicatesses	12.2	63	0
Reggiani Caseifici	20.1	76	0.2
Ricardo Adocicados	15.5	47	0.2
Save-a-lot Markets	2.5	16	0
Victuailles en stock	24.225	78	0.45
Great Lakes Food Market	156.4	50	0.1
Königlich Essen	90.6075	155	0.45
Cactus Comidas para llevar	26.875	27	0
Magazzini Alimentari Riuniti	20.825	40	0
Rattlesnake Canyon Grocery	38	30	0
Split Rail Beer & Ale	35.5	26	0
Trail's Head Gourmet Provision	32.15	9	0
Drachenblut Delikatessen	32.015	32	0
Folk och få HB	12.325	21	0
LILA-Supermercado	27.125	90	0.3
Blondesddsl père et fils	16	50	0
Bon app'	24.5	36	0
Island Trading	24.75	61	0
Rancho grande	172.25	7	0
Tradição Hipermercados	22.375	88	0



- The next step is to define the URL which will be called by your dashboard or data presentation (Flash file) when the XML Data button is pushed. The URL can be any script or application that will return the correctly formatted XML data source. In this example, we are going to use a script called getresults.aspx. This script will connect to the SQL Server database, run a query, and then create an XML file in the correct format. Notice the Excel cell selection icon next to the URL text box, if you want to pass parameters to the script you can use a cell in Excel to concatenate different segments of the URL into a full string.



7. Click the Behavior tab of the XML Data Button properties, and select Refresh on Load. This will run the script when the dashboard (Flash file) is loaded. You can also set the Refresh on Interval to have the script run periodically.

Getresults.aspx Script Definition

The getresults.aspx script used in this example is a very simple example. You can build as much functionality as necessary into the script to make it more interactive or robust. Here is the code:

```
<%@ Page Language="VB" aspcompat=true Debug="true" validateRequest="false" %>

<%

'Get results from database
Dim sSQL as String
Dim oCon as Object
dim oCmd as Object
dim sConnect as String

'Create connection object
sConnect = "Provider=SQLOLEDB;Data Source=sales3;Initial Catalog=Northwind;User
Id=sa;Password=sa;"
oCon = Server.CreateObject("ADODB.Connection")
oCon.ConnectionString = sConnect
oCon.Open

'Select latest totals from the database
dim oRst as Object
oRst = Server.CreateObject("ADODB.RecordSet")
oRst.ActiveConnection = oCon
sSql = "Select Customers.CompanyName as CompanyName, Month(Orders.OrderDate)
as Month, Sum([Order Details].UnitPrice * [Order Details].Quantity) as Revenue FROM
[Order Details], Orders, Products, Customers Where Orders.OrderDate >=
CONVERT(datetime,'01-01-1998') and [Order Details].OrderID = Orders.OrderID and
Orders.CustomerID = Customers.CustomerID Group by Customers.CompanyName,
Month(Orders.OrderDate) Order by Customers.CompanyName,
Month(Orders.OrderDate)"
oRst.Open(sSql)

'Build XML data to return
dim sXML as String
sXML = "<data>"
sXML = sXML & "<variable name=" + chr(34) + "Range0" + chr(34) + ">"

Do Until oRst.EOF
'totalData
sXML = sXML & "<row>"
sXML = sXML & "<column>" & oRst("CompanyName").value & "</column>"
sXML = sXML & "<column>" & oRst("Month").value & "</column>"
sXML = sXML & "<column>" & oRst("Revenue").value & "</column>"
sXML = sXML & "</row>"
```

```

oRst.MoveNext
Loop
sXML = sXML & "</variable>"

'close out the XML data string
sXML = sXML & "</data>"

'Now write the data out so that it gets returned
Response.Write(sXML)

'Cleanup connection vars
oRst.close
oRst = Nothing
oCon.close
oCon = Nothing

%>

```

The first task the script performs is to connect to the database.

```
sConnect = "Provider=SQLOLEDB;Data Source=sales3;Initial Catalog=Northwind;User
Id=sa;Password=sa;"
```

The instance name is sales3 and the database name is Northwind. The Authentication information does not need to be hard coded into the script, it can be passed as parameters, for example from the portal environment the user has already logged into.

The next section of code creates a recordset for the resulting data returned from the database. It then sets the SQL statement and runs the query.

```

oRst = Server.CreateObject("ADODB.RecordSet")
sSql = "Select Customers.CompanyName as CompanyName, Month(Orders.OrderDate)
as Month, Sum([Order Details].UnitPrice * [Order Details].Quantity) as Revenue FROM
[Order Details], Orders, Products, Customers Where Orders.OrderDate >=
CONVERT(datetime,'01-01-1998') and [Order Details].OrderID = Orders.OrderID and
Orders.CustomerID = Customers.CustomerID Group by Customers.CompanyName,
Month(Orders.OrderDate) Order by Customers.CompanyName,
Month(Orders.OrderDate)"
oRst.Open(sSql)

```

The next section of code writes the result set into an XML file. Note the script is returning the information by creating a string that is structured in the correct XML format. There are other ways to create this file, but this is the quickest to return the data in the correct format. A best practices option would be to take the result set and run it through a Style Sheet (XSLT) to transform the data into the correct format. That Style sheet can then be used for all processes that are sending prepared XML to Xcelsius.

```

dim sXML as String
sXML = "<data>"
sXML = sXML & "<variable name=" + chr(34) + "CompanyOrderDetails" + chr(34) + ">"

```

```

Do Until oRst.EOF
totalData

```

```
sXML = sXML & "<row>"
sXML = sXML & "<column>" & oRst("CompanyName").value & "</column>"
sXML = sXML & "<column>" & oRst("Month").value & "</column>"
sXML = sXML & "<column>" & oRst("Revenue").value & "</column>"
sXML = sXML & "</row>"
oRst.MoveNext
Loop
sXML = sXML & "</variable>"
```

Notice that the variable name is defined as the same name that was defined for the ranges in Xcelsius when configuring the XML Data Button.

Testing the Getresults.aspx Script

Copy the script into the root of your Web server. (C:\inetpub\wwwroot). To test just the script, open it in Internet Explorer (IE) by typing <http://localhost/getresults.aspx> in the address bar. If the script works correctly then you will see all the data returned in the IE window. Don't worry about the way the format looks here, if you want to see the true format select View Source from the View menu. Notice the structure of the XML file and match it to the structure of the XML generated from the Preview XML feature in the XML Data Button properties.

Once your script is working properly, you can open the SWF file in IE and test that the data is being refreshed by changing a data value in your database that will be reflected in your table component when it is updated. Then click the Submit button and watch for the change.

Benefits

- Huge level of robustness and flexibility
- Can program any logic into the script
- Provides for the ability to handle Security
- SQL Statement can be dynamically created based on parameters passed to the script.

XML Gotchas

This section will explain other formatting requirements of the XML files:

Data Typing in XML

Xcelsius does not support Data Types within an XML file. It requires all data in XML to be in string format. The only exception to this rule is that date-based data must be in its numeric representation. When pulling data from a database these requirements are usually not an issue, but if another application creates the XML or a user is converting text-based data to a XML, these rules need to be taken into account.

Note: this applies to using the Web Services and the XML Data button connectivity options described in this and other Tech Tip Guides, it does not apply to the Excel 2003 Map option.

Non-Table-Based XML in Excel 2003 Maps

If the data being pulled into Excel Maps is in a non-table-based format, the entire root level of the XML file should be pulled into the spreadsheet instead of picking individual elements. A non-table-based format would be an XML file that lists each customer followed by the Orders/Invoices that belong to the customer. By pulling in the full root level, we allow Excel to keep the data in the structure of the XML file. By pulling elements individually Excel might make some changes so it can display the data the best way it sees fit.

Multiple Range selections in a single XML

For both the Web Services and the XML Data Button connectivity options, it is possible to define multiple ranges of data to be replaced from the same XML file. In this case the structure will be exactly the same, but just repeated within the XML file. Inside the data element there will be two blocks of data each with their own variable names which match the names of the ranges defined in Xcelsius. If there are more ranges than blocks in the XML file, the whole file will not work properly. But, it is possible to have more data in the XML than is needed for the ranges.

----- END -----