

How to

Save and Load the Spreadsheet data

Tools: AltioLive Studio
AltioLive Application Manager
AltioLive Designer

Skill Level: Advanced

Summary: This document explains how to save the data entered in a Spreadsheet and then how to retrieve the data.

Integra SP

88 Wood Street London
EC2V 7RS
United Kingdom

www.altio.com
tel: +44 (0) 20 8528 1045

Contents

Introduction	3
Preparing the database and binding it to the Application.....	5
Preparing your XML database	5
Binding the Database to the application	6
Creating Service Functions from the Application Manager.....	7
Creating a service function to get the data in the back end application	7
Creating the GET_DATA Service Function.....	7
Setting the datakeys properties.....	11
Creating a Service Function to Save the Data of a Spreadsheet.....	11
Creating the User Interface of the Application from the AltioLive Designer.....	17
Launching the Designer	17
Setting the Initial Data Service and Datatypes	17
Adding Controls to the application’s window.....	19
Setting the controls	21

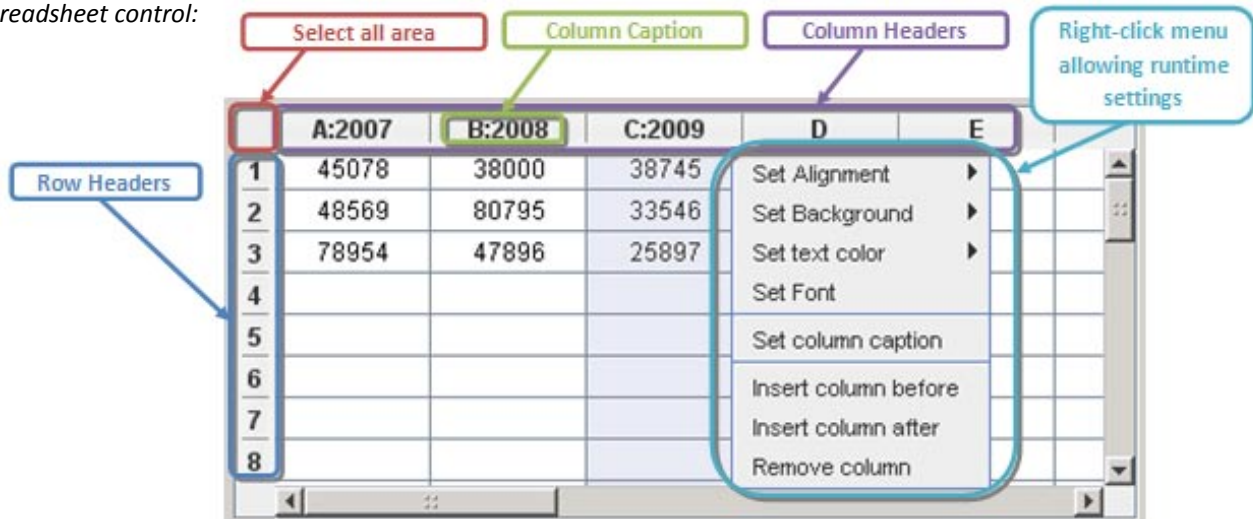
Running a preview of the application.....	25
Creating a drop-down menu to select the available data	26
Loading the saved data back to the spreadsheet	29
Setting the drop-down menu	29
Test your application	31
Saving and Loading the Columns' Names.....	32
Saving the columns' names	32
Loading the saved columns' names	34
Running a preview of the application.....	35
Saving and Loading the Style of the Spreadsheet.....	37
Saving the spreadsheet's style.....	37
Loading the saved style of the spreadsheet	38
Running a preview of the application.....	39

Introduction

The spreadsheet provides an easy way to edit large amounts of static data in a similar way as an Excel spreadsheet. By default the Spreadsheet is empty, but once populated with data you can save these data to your application data base in order to retrieve them the next time you are using your application. This section explains step by step how to:

- create a simple application that saves the data entered in the Spreadsheet (values, columns' names but also styles applied to the spreadsheet)
- load the data back to the spreadsheet.

The spreadsheet control:



The application we are going to create:

The screenshot shows a window titled "Sample window" with a spreadsheet on the left and XML data on the right. The spreadsheet has columns A:2008, B:2009, C:2010, D, E, F, G and rows 1-14. The data in the first three columns is as follows:

	A:2008	B:2009	C:2010	D	E	F	G
1	45278	40785	32497				
2	50650	45789	15000				
3	32457	39045	45786				
4	20785	40146	124796				
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							

Below the spreadsheet, there is a text input field with the placeholder "Enter a name for the Data", a "Save" button, and a dropdown menu labeled "Select the data to load" with "test data" selected.

The XML data on the right is organized into three sections:

- Xml value of the Spreadsheet**: Contains XML for the spreadsheet cells, including values like 45278, 40785, 32497, 50650, 45789, 15000, 32457, 39045, 45786, 20785, 40146, 124796.
- Xml value of the Spreadsheet's Columns**: Contains the column headers: 2008,2009,2010,.....
- Xml value of the Spreadsheet's Style**: Contains XML for cell styles, including background colors (0xD7E3FB) and foreground colors (0x1C3E82).

Preparing the database and binding it to the Application

Preparing your XML database

In this example, the application uses a simple xml database that stores the data contained in the spreadsheet.

This is the XML file we are going to use as a data base for this example:

1. Create a new xml document and copy and paste the following XML in the new file:

```
<EXAMPLE>  
  <MYSPREADSHEET DATA_NAME="example name" SPREAD_VALUE="example value" COLUMN_NAME="example column" STYLE="example style"/>  
</EXAMPLE>
```

Please note: For more clarity, the following convention is followed: a *blue font* is used for all the XML elements, a *red font* for the attributes and a *purple font* for the values of the attributes.

The **<MYSPREADSHEET>** element will hold the data inserted in the spreadsheet. Each data attribute will hold a specific aspect of the spreadsheet contents:

- The **DATA_NAME** attribute will hold the name used to describe the saved data
- The **SPREAD_VALUE** attribute will hold the values entered in the Spreadsheet
- The **COLUMN_NAME** attribute will hold the caption of each column
- The **STYLE** attribute will hold the style applied to the Spreadsheet (fonts, background colors, text colors,...)

Binding the Database to the application

1. From **AltioLive Studio**, create a new project, expand the application node and right-click on the **dbdata** folder.
2. Select the **Upload New File...** option. A window called **File Upload** opens.
3. Browse to select the simple xml file shown above then click on the **Upload file(s)** button.
4. Once the status changes to **1 File(s) Uploaded**, click on the **Close** button.
5. Still from **AltioLive Studio**, right-click on the application node and select the **AltioDB** option. The **AltioDB** window opens.
6. Click on the **Table Data** button to display the **Table Data** window.
7. From the **Table data** window, click on the **Add** button. It should add a new row in the **Tables** list.
8. In the **Tables** list, an empty row should now be displayed. Click in this row to display a drop-down menu.
9. From the drop-down menu, select the xml file you have uploaded then click on the **Close** button.
10. From the **AltioDB Manager** window, click on the **UPDATE** button.
11. An Information window will display, click on **Yes**.
12. A Results window shows that the server has been updated. Click on the **Close** button.
13. Click on the **Reset AltioDB** button to apply the changes to the configuration.
14. An **Information** window displays, click on **Yes**.


The XML data file has now been integrated into the application.

Creating Service Functions from the Application Manager

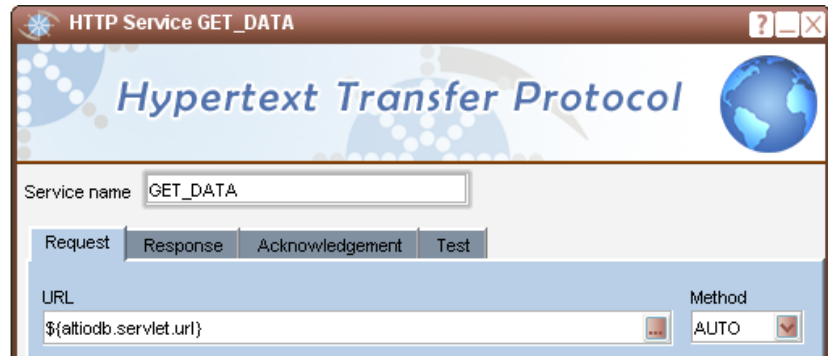
This practical exercise introduces the **Application Manager**. We will add a simple Service Function to retrieve the spreadsheet's data so that it is available when we need it in the Designer and at runtime.

Creating a service function to get the data in the back end application

Creating the GET_DATA Service Function

1. From **AltioLive Studio**, right-click on your project node and select the **Application Manager** option.
2. From the **Application Manager**, click on the **HTTP** button in the menu bar , to create a new HTTP Service Function.
3. Set the Service Function as shown below:

- In the **Service name** field, enter **GET_DATA**.
- In the **URL** field, enter **\${altiodb.servlet.url}**.
It is a parameter that is dynamically replaced with the URL of the Altio DB servlet.



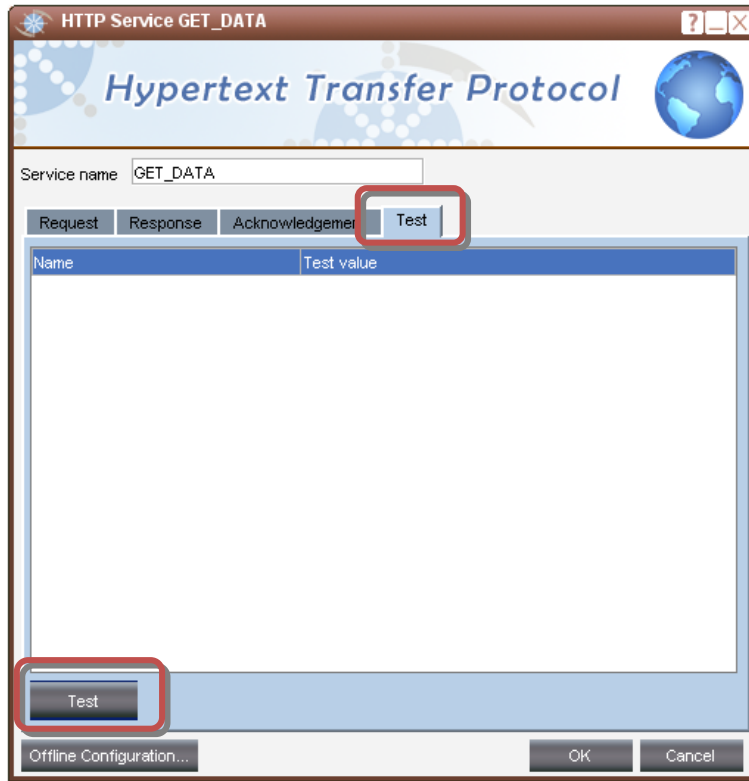
4. Add three parameters and set them as follows:

Parameter mapping	
Name	Value
APPID	\${application.id}
ACTION	GET
TARGET	/EXAMPLE

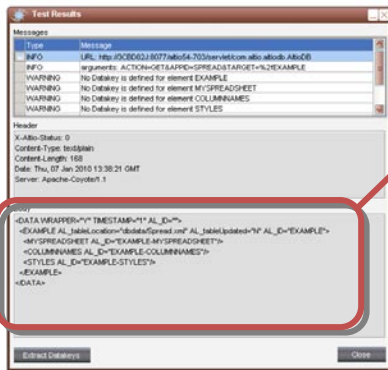
- **APPID:** the value **\${application.id}** is a parameter that is dynamically replaced with the application's ID from the session.
- **ACTION:** the value **GET** means that this service gets data from the server (as opposed to updating or deleting). It is a fix string required by the **Altio DB Manager**.
- **TARGET:** the value **/EXAMPLE** indicates the root of required data. It means that all of the data under the **EXAMPLE** data element will be returned. Other back-end applications will require different parameters, since these are specific to Altio DB.

The Service Function to retrieve the data is set. We just have to test it to ensure it works properly.

5. Go to the **Test** tab then click on the **Test** button.



6. Shortly afterwards the **Test Results** window opens. It should display the content of our XML file in the **Body** frame:



Body

```
<DATA WRAPPER="Y" TIMESTAMP="1" AL_ID="">
  <EXAMPLE AL_tableLocation="dbdata/Spread.xml" AL_tableUpdated="N" AL_ID="EXAMPLE">
    <MYSPREADSHEET DATA_NAME="example name" SPREAD_VALUE="example value"
      COLUMN_NAME="example column" STYLE="example style" AL_ID="EXAMPLE-MYSPREADSHEET"/>
  </EXAMPLE>
</DATA>
```


Please note: The **WARNING** entries in **Messages** are displayed because no **datakeys** have been defined for the elements in the data returned. **Datakeys** are used to uniquely identify each data element within AltioLive. A datakey is defined for an element name and optionally a namespace. The datakey is then used by AltioLive to create the special attribute **AL_ID**. In the data above you will see that each element has been given a default datakey which is not unique.

7. Click the **Extract Datakeys** button to create datakey definitions for the **EXAMPLE** and **MYSPREADSHEET** elements.
8. When the **Extract Datakeys** window displays, click **OK**.
9. Close the **Test Results** window.

Setting the datakeys properties

1. From the **Application Explorer** window, expand the **Datakeys** node.
2. Display the **Properties** window by clicking on **View | Properties** menu item.
3. Select the **MYSPREADSHEET** datakey and from the **Properties** window:
 - Enter **MS-** in the **Prefix** field.
 - Enter **@DATA_NAME** in the **Key** field.

Please note: The **EXAMPLE** datakey can be ignored as there will be no multiple instances of this element.

4. Click on the **Save** icon . Click **OK** in the new window to confirm that the configuration should be updated and the application restarted. A **Validation Results** window will report that the server has been updated.

Please note: If your application contains errors, it will not be saved unless you tick the **Save even when it is invalid** option from the **Confirm Save** window.

Creating a Service Function to Save the Data of a Spreadsheet

We have created a Service Function to get the data from the XML database. We are going to create a Service Function to save the data entered in the Spreadsheet at runtime. It will get the data and add it to the XML database as attributes of the **<MYSPREADSHEET>** elements. To create this Service Function:

1. Since we are going to the same URL, the easiest way to create the Service Function is to copy the existing one: From the **Application View** window, expand the **Services** node and right-click on the **GET_DATA** Service Function. Select **Clone** in the context menu.
2. Open the new service function, called **GET_DATA1**, by double-clicking on it.
3. In the **Service name** field, enter the name of the new Service Function: **SAVE_DATA**.
4. On the **Request** tab, change the **ACTION** parameter to **NEW**. The value **NEW** means that this service creates a new data element. It is a fix string required by the **Altio DB Manager**.
5. In addition to the existing parameters, create five new parameters and set them as follows:
 - **AL_NAME: MYSPREADSHEET**. We provide the name of the new element to create, in this example: **MYSPREADSHEET**.
 - **DATA_NAME: \${client.DATA_NAME}** This provides the name of the element's attribute to create. In this example the Service Function will create a **DATA_NAME** attribute for the **MYSPREADSHEET** element. The value of the attribute is set to **\${client.DATA_NAME}** meaning it will be provided at runtime.
 - **SPREAD_VALUE: \${client.SPREAD_VALUE}**. This provides the name of the element's attributes to create. In this example it will create a **SPREAD_VALUE** attribute for the **MYSPREADSHEET** element. The value of the attribute is set to **\${client.SPREAD_VALUE}** meaning it will be provided at runtime.

- **COLUMN_NAME:** `$(client.COLUMN_NAME)` This provides the name of the element's attributes to create. In this example it will create a **COLUMN_NAME** attribute for the **MYSPREADSHEET** element. The value of the attribute is set to `$(client.COLUMN_NAME)` meaning it will be provided at runtime.
- **STYLE:** `$(client.STYLE)` This provides the name of the element's attributes to create. In this example it will create a **STYLE** attribute for the **MYSPREADSHEET** element. The value of the attribute is set to `$(client.STYLE)` meaning it will be provided at runtime.

All the Parameters of the **SAVE_DATA** Service Function:

Parameter mapping	
Name	Value
APPID	<code>\$(application.id)</code>
ACTION	NEW
TARGET	/EXAMPLE
AL_NAME	MYSPREADSHEET
DATA_NAME	<code>\$(client.DATA_NAME)</code>
SPREAD_VALUE	<code>\$(client.SPREAD_VALUE)</code>
COLUMN_NAME	<code>\$(client.COLUMN_NAME)</code>
STYLE	<code>\$(client.STYLE)</code>

6. On the **Acknowledgement** tab, set the following parameters:

ACKNOWLEDGEMENT TAB		Description
Source	HEADER	The Altio Presentation Server searches for a header parameter named X-Altio-Status that specifies whether the service has failed or succeeded.
On success: Source	DEFINE	Option that allows to enter a text to send when a service succeeds.
On success: Text	Data saved successfully.	This is the message that will be sent to the AltioLive client when a service succeeds.
On failure: Source	DEFINE	Option that allows to enter a text to send when a service fails.
On failure: Text	It was not possible to save these data.	This is the message that will be sent to the AltioLive client when a service fails.

7. On the **Test** tab, test the service function by:
 - a. adding test values in the **Test value** field. For example:


Request	Response	Acknowledgement	Test
Name		Test value	
client.SPREAD_VALUE		test data	
client.DATA_NAME		test data	
client.STYLE		test data	
client.COLUMN_NAME		test data	

- b. pressing the **Test** button.

This should produce a result similar to the XML below (although some of the values depend on your test values):

```
<DATA AL_ID="">
  <EXAMPLE AL_tableLocation="dbdata/Spread.xml" AL_tableUpdated="Y" AL_ID="EXAMPLE">
    <MYSPREADSHEET STYLE="test data" SPREAD_VALUE="test data" COLUMN_NAME="test data" DATA_NAME="test data" AL_TS="2" AL_ID="MS-test data"/>
  </EXAMPLE>
</DATA>
```

8. Close the **Test Results** window.
9. Press **OK** to accept the service function.



10. Click on the **Save** icon . Click **OK** in the new window to confirm that the configuration should be updated and the application restarted. A Results window will report that the server has been updated.
11. Provided the server was updated successfully, close the web browser's window running the **Application Manager**.

Creating the User Interface of the Application from the AltioLive Designer

The **Designer** provides a suite of tools to create the interface an end-user will see. We will add a spreadsheet to a window as well as a few other controls to allow a user to save the data he has entered in the Spreadsheet.

Please note: To create and set the user interface, you will need the **View Explorer** and the **Properties** windows. To display them, simply click on **View | Explorer** and **View | Properties** menu items.

You can also display them with their icons displayed in the menu bar:

-  , to display the **View Explorer** window,
-  , to display the **Properties** window.

Launching the Designer

1. From **AltioLive Studio**, right-click on your project and select **View Designer** in the context menu.

The **Designer** is started in a new browser window.



Setting the Initial Data Service and Datatypes

Earlier in the session we created a back-end application using the **AltioDB Manager**. This accesses a static XML data file. Then, we created a Service Function (**GET_DATA**) in the **Application Manager**, to provide data from the back end application.

To complete the link between the data and our user interface, we need to add a call to the Service Function. This can be done in the **Designer**.



1. From the **View Explorer** Window, right-click on **Initial Data Services** and select **New Initial Data** in the contextual menu.
2. A new node, called **INCLUDE**, should be displayed. Select it.
3. From the **Properties** window, expand the **General** Properties.
4. From the **Server Command** drop-down menu, select **GET_DATA**.

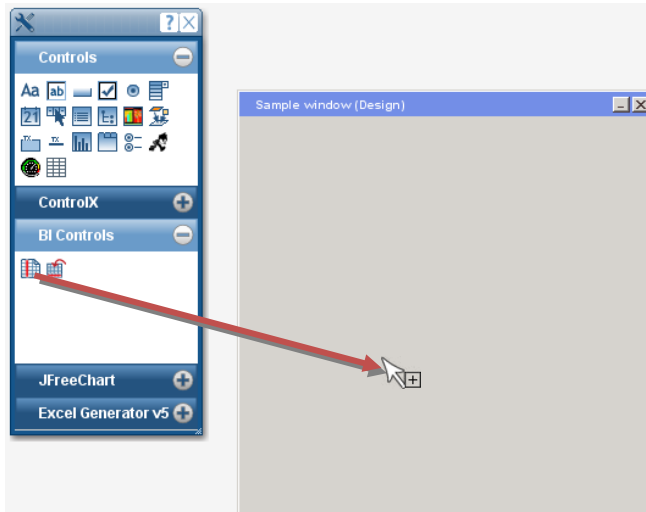
When you make the selection, the Service Function is called by the **Designer** and the XML data is made available.




5. Click the **Show Datatypes** icon  to open the **Datatypes** window.
6. Click on the **Import from Service...** icon  located on the left side of the **Datatypes** window. The **Import datatype information** window is then displayed.
7. From the **Service Function** drop-down menu, select **GET_DATA** and click on the **Import** button.
8. The **Datatypes** window should now display the **EXAMPLE** element under the **DATA** node.
9. Expand the **EXAMPLE** and **MYSPREADSHEET** nodes to display the available attributes.

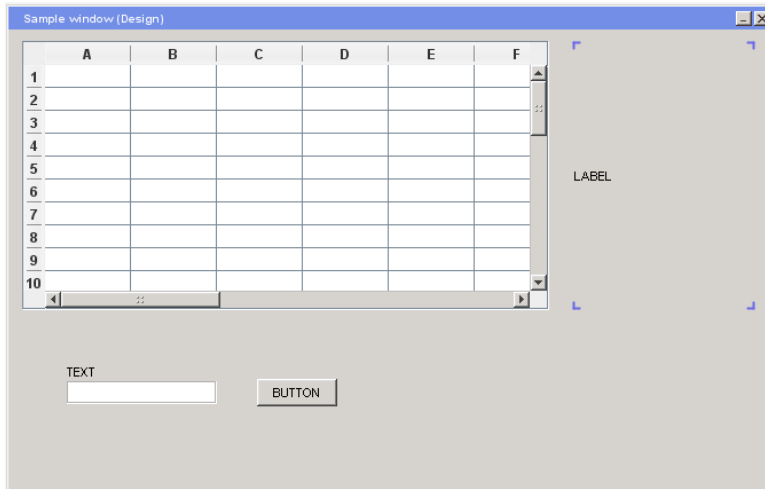
Now the link between the data and the user interface is fully completed.

Adding Controls to the application's window.

1. From the **View Explorer** window, expand the **Windows** node. By default, a window is already created.
2. To display a preview of this window, double-click on it.
3. We now need to show the **Control Palette** in order to add a spreadsheet control to this window. To do so, click on the **Show Control Palette** icon .
4. From the **Control Palette**, expand the **BI Controls** panel. The spreadsheet and datagrid controls are now visible.
5. Drag the spreadsheet icon  from the **Control Palette** and drop it onto the window:



6. A Spreadsheet should now be added to the window.
7. In order to save the spreadsheet data, we need the user to be able to enter a name to describe the entered data then to click on a save button. Add the following controls:
 - a text field  (where the user will enter the name he wants to give to the data)
 - a button  (which will be clicked by the user to save the data).
 - For more clarity, we can also add a label  that will display the data of the spreadsheet converted in XML. The data will be sent to the database.
8. Reposition and resize the controls so that the window looks like the window shown below.



Setting the controls

We now need to set these controls so that when the user has entered some data in the spreadsheet he can give a name to the data and save it to the data base.


1. From the **View Explorer**, select the spreadsheet.
2. From the **Properties** window, set the spreadsheet as follows:

	Property	Value	Comments
▼	General		
	Name	SPREADSHEET	

3. From the **View Explorer**, select the text field.
4. From the **Properties** window, set the text field as follows:


	Property	Value	Comments
▼	General		
	Name	DATA_NAME	As this text field is meant to provide the value of the DATA_NAME attribute then it has to have the same name as the attribute.
	Caption	Enter a Name for the Data	

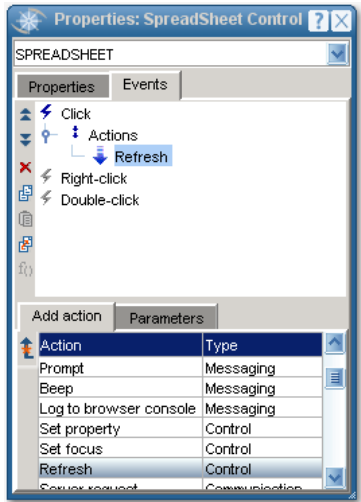
5. From the **View Explorer**, select the label.
6. From the **Properties** window, set the label as follows:

Property	Value	Comments
▼ General		
Name	SPREAD_VALUE	As this label is meant to provide the value of the SPREAD_VALUE attribute then it has to have the same name as the attribute.
Caption	Xml value of the Spreadsheet	
Datafield	<code>\${SPREADSHEET.datagriddataxml}</code>	This is set to display the value of the spreadsheet in XML. The label uses one of the runtime properties of the spreadsheet (datagriddataxml). All the available runtime properties can be shown in the Syntax Builder. The latter can be accessed by clicking on the ellipsis button  in the Data Field property.

The Label control needs further settings, as the data of the spreadsheet are going to change every time the user edits it; the value of the label needs to be refreshed to display the latest values of the Spreadsheet. To do so:

7. From the **View Explorer**, select the Spreadsheet.
8. From the **Properties** window, go to the **Events** tab. All the available events for this control are displayed.
9. Select the **Click** event and, in the list below it, select the **Refresh** action.

10. Click on the **Add action** icon . You should get a window similar to the one below:




11. Now click on the **Parameters** tab and set the **Control** property to the Label, called **SPREAD_VALUE**.

This means that every time the user will click on the spreadsheet, the label will be refreshed and will display the latest data entered in the spreadsheet.

12. From the **View Explorer**, select the button.
13. From the **Properties** window, set the button as follows:

	Property	Value	Comments
▼	General		
	Name	SAVE_BUTTON	
	Caption	Save	


The Save button needs further settings as we need to call the **SAVE_DATA** Service Function when the user clicks on the button. To do so:

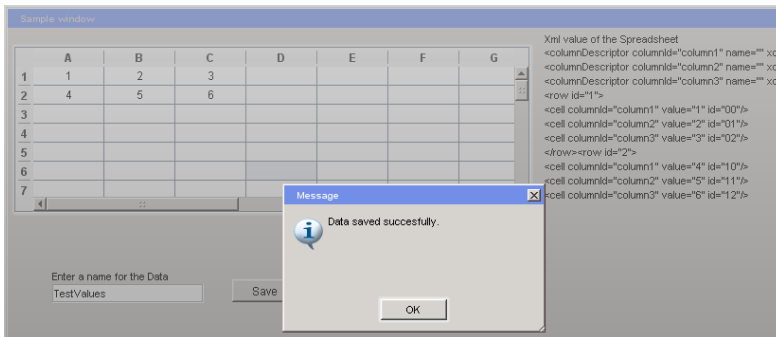
14. Still with the button selected, from the **Properties** window, go to the **Events** tab.
All the available events for this control are displayed.
15. Select the **Click** event.
16. From the **Add action** tab a list displays all the available events for this control. Select the **Server Request** action.
17. Click on the **Add action** icon .
18. Now click on the **Parameters** tab and set the parameters as follows:

	Parameter	Value	Comments
▼	Parameter	Value	
	Service Function	SAVE_DATA	The Service Function we created earlier.
	Parameter Source	WINDOWDATA	Sends the value associated with each control on the window.

Running a preview of the application

You can now click on the **Run** button to see a preview of your application and ensure that it works properly.

1. Click on the Run icon  displayed in the menu bar.
2. Enter some data in the spreadsheet.
3. Click in one of the cells of the spreadsheet to ensure the Label refreshes and displays the XML values of the Spreadsheet.
4. Enter a name to describe the data in the text field.
5. Click on the **Save** button. A message window should pop up reading “Data saved successfully”.

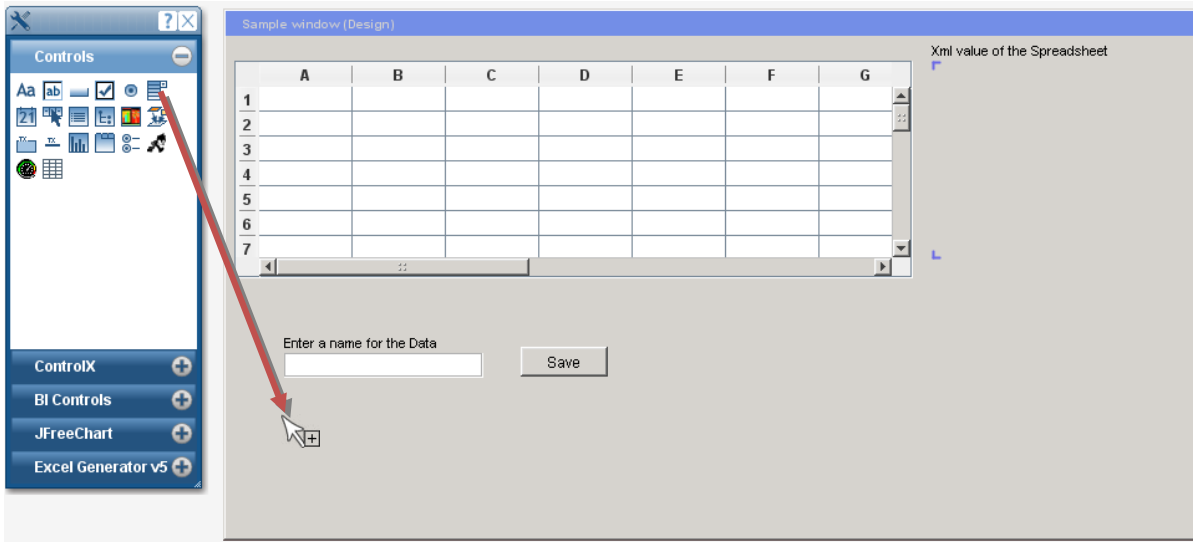


6. Click on the **Stop**  button to go back to the Designer.

Creating a drop-down menu to select the available data



We can now add a drop-down menu to the window to display all the saved data. This is going to be a data-driven control: The select control will have its column populated from our XML data file.

1. From the Control Palette , drag a select control  and drop it onto the window, under the text field.

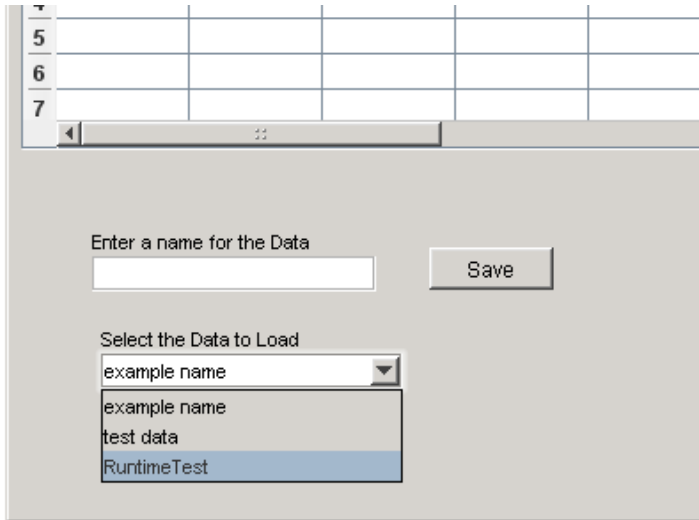


- From the **View Explorer**, select the select control.
- From the **Properties** window, set the select control as follows:

Property	Value	Comments
▼ General		
Name	DATA_SELECT	
Caption	Select the Data to Load	
List data source	EXAMPLE/MYSPREADSHEET	Xpath to the element holding the saved data sets of the spreadsheet.
List data field	@DATA_NAME	This will use the values held in the DATA_NAME attributes to create a list of all the data sets available.
List data value	@DATA_NAME	This will use the values held in the DATA_NAME attribute to create a list of all the data sets available.

- Save your application  then run it .
- Enter some data in the spreadsheet.
- Click in one of the cells of the Spreadsheet to ensure the Label refreshes and displays the latest XML values of the Spreadsheet.

7. Enter a name to describe the data in the text field. For example “Runtime Test data”.
8. Click on the **Save** button.
9. A message window should pop up reading “Data saved successfully”. Click on **OK**.
10. Now expand the drop-down menu: all the names of the saved data should be displayed, including the one you have just saved (in our example “Runtime Test data”).




11. Click on the Stop  button to go back to the Designer.

Loading the saved data back to the spreadsheet

Setting the drop-down menu

We already have all the Service Functions we need to save and load the data to the spreadsheet. We just need further settings to make the spreadsheet display the data selected in the drop-down menu.

1. From the **View Explorer**, select the select control.
2. From the **Properties** window, go to the **Events** tab.
3. Select the **Change** event and, in the list displayed in the **Add action** tab, select the **Set Property** action.
4. Click on the **Add action** icon .

This means every time the data selected in the drop-down menu is changed, it is going to trigger the **Set Property** action. We are going to parameter this action so that it sets the Data source property of the Spreadsheet to the selected data.

5. Click on the **Parameters** tab and set the parameters as follows:




Parameter	Value	Comments
▼ Parameter	Value	
Control	SPREADSHEET	It specifies which control to set.
Property	DATAGRID_XML	It specifies which property to set.
Value	eval(/EXAMPLE/MYSPREADSHEET[@DATA_NAME=\${DATA_SELECT.value}]/@SPREAD_VALUE)	It specifies which value to set for the property selected above.

Here are more explanations about the complex Xpath used in the **Value** property; we cut it into sections to explain step by step what this Xpath predicate selects. This Xpath will select only:

- **/EXAMPLE/MYSPREADSHEET:**
The **MYSPREADSHEET** elements that are children elements of **EXAMPLE** elements.
- **MYSPREADSHEET[@DATA_NAME=\${DATA_SELECT.value}]:**
This **MYSPREADSHEET** element needs to have a **DATA_NAME** attribute that is equal to the value selected in the **DATA_SELECT** control. There should be only one **MYSPREADSHEET** element that has a **DATA_NAME** attribute that is equal to the value selected in the **DATA_SELECT** control.
- **/@SPREAD_VALUE:** Once this element is identified, then we use the value of its **SPREAD_VALUE** attribute to set the **DATAGRID_XML** property of the Spreadsheet.

```
<EXAMPLE>
  <MYSPREADSHEET DATA_NAME="example name"
    SPREAD_VALUE="example value"
    COLUMN_NAME="example column" STYLE="example
    style"/>
  <MYSPREADSHEET DATA_NAME="test data"
    SPREAD_VALUE="test data" COLUMN_NAME="test data"
    STYLE="test data"/>
  <MYSPREADSHEET DATA_NAME="Runtime Test data"
    SPREAD_VALUE="&lt;table
    name="&quot;Datagrid&quot;/&gt;&#13;&#10;"
    COLUMN_NAME="" STYLE="" />
</EXAMPLE>
```

Test your application

1. Save your application  then run it .
2. Enter some data in the spreadsheet. Ensure the Label refreshes and displays the XML values of the Spreadsheet.
3. Enter a name to describe these data in the text field. For example “Test data2”.
4. Click on the **Save** button.
5. A message window should pop up reading “Data saved successfully”. Click on **OK**.
6. Now expand the drop-down menu: all the names of the saved data should be displayed, including the one you have just saved (in our example “Test data2”).
7. Still from the drop-down menu, select one of the previously saved data (i.e. in our example “Runtime Test data”).
8. Once selected, the data should be loaded to the Spreadsheet.
9. Click on the **Stop**  button to go back to the Designer.


Saving and Loading the Columns' Names

The way to save and load the columns' captions of the spreadsheet is very similar to the way we proceeded to save and load the contents of the spreadsheet. To save the columns' caption, instead of getting the XML value of the spreadsheet, we are going to get the xml value of the columns' captions. And instead of sending these data to the **SPREAD_VALUE** attribute of the **MYSPREADSHEET** element, we are going to send these data to the **COLUMN_NAME** attribute of the **MYSPREADSHEET** element.

Saving the columns' names

We are going to add another label to display the xml value of the columns' names. We will then send these XML data to the database. These data are going to be stored in the **COLUMN_NAME** attribute of the **MYSPREADSHEET** element.

1. Add a **Label** under the existing one.
2. From the **Properties** window, set it as follows:

Property	Value	Comments
▼ General		
Name	COLUMN_NAME	As this label is meant to provide the value of the COLUMN_NAME attribute then it has to have the same name as the attribute.
Caption	Xml value of the Spreadsheet's Columns	
Datafield	\${SPREADSHEET.columnncaptions}	This is set to display the value of the spreadsheet's columns in XML. The label uses one of the runtime properties of the spreadsheet (columncaption). All the available runtime properties can be shown in the Syntax Builder that can be accessed by clicking on the ellipsis button  in the Data Field property.

The Label control needs further settings, as the captions of the columns are going to change every time the user edits them; the value of the label needs to be refreshed to display the latest columns' caption of the Spreadsheet. We are going to use a similar approach to what had been done for the **SPREAD_VALUE** label. To do so:

3. From the **View Explorer**, select the Spreadsheet.
4. From the **Properties** window, go to the **Events** tab.
5. Expand the **Click** event as well as the **Actions** node under it. The **Refresh** action for the **SPREAD_VALUE** label should be displayed.
6. Right-click on the **Refresh** action and select **Clone**.
7. Select the cloned **Refresh** action and from the **Control** drop-down menu, select the **COLUMN_NAME** label.

This means that every time the user will click on the spreadsheet, both **SPREAD_VALUE** and **COLUMN_NAME** labels will be refreshed and will display the latest data entered in the spreadsheet.

Loading the saved columns' names


We just need to add a **Set property** action to the drop-down menu. We will proceed as we did to load the content of the spreadsheet: once the user selects the name of the data that he wants to load, it sets the spreadsheet's properties to display the contents and columns' captions related to the selected data's name.


1. From the **View Explorer**, select the **DATA_SELECT** drop-down menu.
2. From the **Properties** window, go to the **Events** tab.
3. Expand the **Change** event as well as the **Actions** node under it. The **Set property** action should be displayed.
4. Right-click on the **Set Property** action and select **Clone**.
5. Select the cloned **Set property** action and set it as follows:

▼	Parameter	Value	
	Control	SPREADSHEET	It specifies which control to set.
	Property	COLUMN_CAPTIONS	It specifies which property to set.
	Value	eval(/EXAMPLE/MYSPREADSHEET[@DATA_NAME=\${DATA_SELECT.value}]/@COLUMN_NAME)	It specifies which value to set for the property selected above. Here it will be the value of the COLUMN_NAME attribute held in the MYSPREADSHEET element that has a DATA_NAME attribute matching the value selected in the DATA_SELECT drop-down menu.

Running a preview of the application

You can now click on the **Run** button to see a preview of your application and ensure that it works properly.

1. Click on the Run icon  displayed in the menu bar.
2. Right-click on a column's header and select **Set column caption**.
3. From the **Input window**, enter some caption then click on **OK**.
4. Enter some data in the spreadsheet.
5. Click in one of the cell of the Spreadsheet to ensure the labels refresh and display the XML values of the Spreadsheet as well as the XML columns' caption.
6. Enter a name to describe the data in the text field (for example "Test data3").
7. Click on the **Save** button.
8. A message window should pop up reading "Data saved successfully". Click on **OK**.
9. Now expand the drop-down menu: all the names of the saved data should be displayed, including the one you have just saved (in our example "Test data3").
10. Still from the drop-down menu, select one of the previously saved data (i.e. in our example "Test data2").
11. Once selected, the data should be loaded to the Spreadsheet and, as no columns' captions were saved with the data, the column's headers should only display letters).

12. Then select back the data you have just saved (in our example “Test data3”).
13. Once selected, the data should be loaded to the Spreadsheet as well as the columns’ captions.
14. Click on the **Stop**  button to go back to the Designer.

Saving and Loading the Style of the Spreadsheet

We are again going to proceed the same way to save and load the spreadsheet's style: instead of getting the XML value of the spreadsheet or its columns captions, we are going to get the xml value of the style of the spreadsheet. And instead of sending the data to the **SPREAD_VALUE** attribute of the **MYSPREADSHEET** element, we are going to send the data to the **STYLE** attribute of the **MYSPREADSHEET** element.

Saving the spreadsheet's style

We are going to add another label to display the xml value of the spreadsheet's style. We will then send these XML data to the database. The data will be stored in the **STYLE** attribute of the **MYSPREADSHEET** element.

1. Add a Label under the drop-down menu.
2. From the Properties window, set it as follows:

General			
	Name	STYLE	As this label is meant to provide the value of the STYLE attribute then it has to have the same name as the attribute.
	Caption	Xml value of the Spreadsheet's Style	
	Datafield	`\${SPREADSHEET.styledataxml}`	This is set to display the value of the spreadsheet's style in XML. The label uses one of the runtime properties of the spreadsheet (styledataxml). All the available runtime properties can be shown in the Syntax Builder.

The Label control needs further settings, as its value needs to be refreshed to display the latest xml style of the Spreadsheet. We are going to use a similar approach to what had been done for the **SPREAD_VALUE** and **COLUMN_NAME** labels:

3. From the **View Explorer**, select the Spreadsheet.
4. From the **Properties** window, go to the **Events** tab.
5. Expand the **Click** event as well as the **Actions** node under it. The **Refresh** action for the **SPREAD_VALUE** and **COLUMN_NAME** labels should be displayed.
6. Right-click on one of the **Refresh** action and select **Clone**.
7. Select the cloned **Refresh** action and from the **Control** drop-down menu, select the **STYLE** label.

This means that every time the user will click on the spreadsheet, **SPREAD_VALUE**, **COLUMN_NAME** and **STYLE** labels will be refreshed and will display the latest data entered in the spreadsheet.

Loading the saved style of the spreadsheet

We just need to add a **Set property** action to the drop-down menu. We will proceed as we did to load the content of the spreadsheet: once the user selects the name of the data that he wants to load, it sets the spreadsheet's properties to display the contents, columns' captions and styles related to the selected data's name.


1. From the **View Explorer**, select the **DATA_SELECT** drop-down menu.

- From the **Properties** window, go to the **Events** tab.
- Expand the **Change** event as well as the **Actions** node under it. Two **Set property** actions should be displayed.
- Right-click on one of the **Set Property** action and select **Clone**.
- Select the cloned **Set property** action and set it as follows:

Parameter	Value	Comments
Parameter	Value	
Control	SPREADSHEET	It specifies which control to set.
Property	DATAGRID_STYLE	It specifies which property to set.
Value	<code>eval(/EXAMPLE/MYSPREADSHEET[@DATA_NAME=\${DATA_SELECT.value}]/@COLUMN_NAME)</code>	It specifies which value to set for the property selected above. Here it will be the value of the STYLE attribute held in the MYSPREADSHEET element that has a DATA_NAME attribute matching the value selected in the DATA_SELECT drop-down menu.

Running a preview of the application

You can now click on the **Run** button to see a preview of the application and ensure that it works properly.

- Click on the Run icon  displayed in the menu bar.
- Right click on a column's header and select **Set column caption**.

8. From the **Input window**, enter some caption then click on **OK**.
9. Change the Style of the spreadsheet by right-clicking on the spreadsheet and using the following options: **Set alignment, Set Background, Set text color, Set font**.
10. Enter some data in the spreadsheet.
11. Click in one of the cells of the spreadsheet to ensure the labels refresh and display the latest XML values of the Spreadsheet.
12. Enter a name to describe the data in the text field (for example “Test data4”).
13. Click on the **Save** button.
14. A message window should pop up reading “Data saved successfully”. Click on **OK**.
15. Now expand the drop-down menu: all the names of the saved data should be displayed, including the one you have just saved (in our example “Test data4”).
16. Still from the drop-down menu, select one of the previously saved data (i.e. in our example “Test data3”).
17. Once selected, the data should be loaded to the Spreadsheet.
18. Select back the data you have just saved (in our example “Test data4”).
19. Once selected, the data should be loaded to the Spreadsheet as well as the column’s captions and the spreadsheet style.

Document Information

Integra SP – Altio

Telephone: +44 (0) 20 8528 1045

Internet: www.altio.com

Copyright © 2010 Integra SP

Copyright in this document is vested in Integra SP. The contents of the document (wholly or in part) must not be reproduced, distributed, used or disclosed without the prior written permission of Integra SP.

Integra recognizes the trademarks or registered trademarks of any third party product or company name referenced in this document at the time of its publication.