

# webElements 4.33

## User Guide

---

### Overview

webElements is a library of custom functions which allows a report developer to embed web controls into a Crystal report.

webElements controls can be used as an alternative to the default prompt dialogue / panel.

webElements controls can be used to add additional functionality or interactivity to a report.

webElements controls can also be used to create a user interface to perform database write-backs, via a target report based on a stored procedure.

This document describes how to install the webElements custom function suite on an SAP BI / BusinessObjects Enterprise system.

Each of the functions used by a report developer, and their required parameters, are described.

A troubleshooting section can be found at the end of the document that will help diagnose and repair common problems.

# Contents

<b>OVERVIEW</b> .....	<b>1</b>
<b>CONTENTS</b> .....	<b>2</b>
<b>INTRODUCTION</b> .....	<b>3</b>
<b>USING WEBELEMENTS IN A CRYSTAL REPORT</b> .....	<b>4</b>
<i>Configuring the BI server to allow pass-through HTML</i> .....	4
<i>Installing webElements</i> .....	5
<i>Removing webElements</i> .....	6
<i>Requirements for all Crystal Reports</i> .....	6
<i>Report Creation with webElements</i> .....	7
<b>HOW TO EXAMPLE</b> .....	<b>8</b>
<i>Creating a report that includes webElements</i> .....	8
<i>Add a select menu to a report</i> .....	8
<i>Pass a prompt to a webElements control</i> .....	9
<i>Add a submit button to a report</i> .....	9
<b>WEBELEMENTS FUNCTION REFERENCE</b> .....	<b>11</b>
<i>Quick Reference</i> .....	11
<i>Main Elements</i> .....	18
<i>Admin</i> .....	20
<i>Admin &gt; Source</i> .....	20
<i>Buttons and Links</i> .....	21
<i>Formatting</i> .....	25
<i>Form Controls</i> .....	31
<i>Inputs and Controls</i> .....	36
<i>Menus</i> .....	60
<i>Other</i> .....	61
<b>TROUBLESHOOTING</b> .....	<b>63</b>
<b>TIPS AND TRICKS</b> .....	<b>65</b>
<b>FINDING MORE INFORMATION</b> .....	<b>66</b>

## Introduction

This Custom Function suite allows report developers to add dynamic web controls to Crystal Reports

- Add custom parameter screens for Crystal Reports
- Improve the functionality of your reports with interactive controls, such as check-boxes, select menus, or calendars.

PLANT PICKER

Source Filters: Renewable (All, None, Wind, Solar, Biomass, Hydro, Geotherm), Nonrenewable (None, Oil, Coal, Gas, Nuclear, Other Fossil, Other)

Geographical Filters: State > County (All)

Outputs: Net Generation (GWh) from -2 to 27, CO<sub>2</sub> Equivalents (tons) from 0 to 30000000

EPA Regional Filter: NERC > Subregion (All), Alaska Systems Coordinating Council, Florida Reliability Coordinating Council, Hawaiian Islands Coordinating Council, Midwest Reliability Organization, Northeast Power Coordinating Council, ReliabilityFirst Corporation, Southeastern Electric Reliability Council

Update Plant List, Reset Report

Use the column heading 'arrows' to sort the report. Right click on any line below and choose 'open in new tab or window' to look at detailed plant level information.

Number of plants based on filter criteria = 304.

Plant Name	Plant ID (ORISPL)	State / County	Primary Fuel	Output (GWh)			Annual CO <sub>2</sub> Equivalent Emissions	
				Net Generation	Renewable Sources	Nonrenewable Sources	Mass (tons)	Output Rate (lb/MWh)
85 A	52162	California / Kern	Wind	0.016	0.016	-	0.00	0.00
85 B	52163	California / Kern	Wind	0.023	0.023	-	0.00	0.00
Adams Wind Farm	56307	Minnesota / Mower	Wind	0.069	0.069	-	0.00	0.00
Aeolus Wind Facility	56225	Texas / Hansford	Wind	0.0059	0.0059	-	0.00	0.00
Aeroturbine	56214	California / Riverside	Wind	0.0071	0.0071	-	0.00	0.00
Agassiz Beach LLC	55576	Minnesota / Clay	Wind	0.0057	0.0057	-	0.00	0.00
AgriWind	56408	Illinois / Bureau	Wind	0.0038	0.0038	-	0.00	0.00
Ainsworth Wind	56162	Nebraska / Brown	Wind	0.18	0.18	-	0.00	0.00
Allegheny Ridge Wind Farm	56451	Pennsylvania / Cambria	Wind	0.064	0.064	-	0.00	0.00
Alendorf	56215	Iowa / Osceola	Wind	0.00096	0.00096	-	0.00	0.00
Altamont Midway Ltd	50001	California / San Joaquin	Wind	0.017	0.017	-	0.00	0.00
Altamont Pass Windplant	50712	California / Alameda	Wind	0.68	0.68	-	0.00	0.00
Altech	50619	California / San Joaquin	Wind	0.014	0.014	-	0.00	0.00
Altech II	50485	California / Riverside	Wind	0.049	0.049	-	0.00	0.00
Aragonne Wind LLC	56336	New Mexico / Guadalupe	Wind	0.24	0.24	-	0.00	0.00
Autumn Hills LLC	55566	Minnesota / Lincoln	Wind	0.0058	0.0058	-	0.00	0.00
B&K Energy Systems LLC	56210	Minnesota / Jackson	Wind	0.0066	0.0066	-	0.00	0.00

Creating a web-based form is done by creating a function-driven formula in Crystal Reports

1. Add webElements to your report from your repository.
2. Drag and drop required webElements into a Crystal formula.
3. Fill in parameter values of the webElements functions.
4. Add the path of the target form to the formula.
5. Publish the report to the Enterprise or Crystal Server environment.

Report Developers do not need any HTML programming knowledge

- The webElements functions automatically generate the required HTML and JavaScript.
- webElements are Crystal Custom Functions and therefore use input methods identical to existing Crystal Reports functions.
- Makes use of the Business Intelligence tools that you already have. With Crystal Reports and either SAP BI / BusinessObjects Enterprise or Crystal Reports Server report designers can create interactive reports and web-based forms.
- **For easier report creation, copy and paste formulae (that have default syntax) from the webElements\_433.rpt onto your report**

## Using webElements in a Crystal Report

To use webElements in a Crystal report you must have access to a SAP BI / BusinessObjects Enterprise or Crystal Reports Server system that is configured to allow pass-through html.

### Configuring the BI server to allow pass-through HTML

When pass-through HTML is enabled on your BusinessObjects Enterprise (BOE) Server or Crystal Reports Server, it is possible to write HTML code that is not pre-processed by the system. This allows a user to create a function that will output HTML tags that will be treated as HTML by the client browser. Normally, if a function outputs HTML tags, those tags are processed by the server and rendered as plain text on the client browser.

By default, pass-through HTML is disabled for security reasons.

**It is recommended that the System Administrator change the default behaviour of the Enterprise system.**

#### To enable pass-through HTML for an Enterprise BI environment

1. Go to the “custom config” folder in your Enterprise BI system. For example:

```
<BOE_Install_Dir>\<Enterprise_DIR>\warfiles\webapps\BOE\WEB-INF\config\custom
```

2. Create a new file named “CrystalReports.properties”.
3. Add the following to the new “properties” file.

```
# This parameter is used to activate pass-through HTML for Crystal Reports  
crystal_encode_html_for_single_line_field_objects=no
```

4. Run wdeploy.

## Installing webElements

webElements can be installed from a report to a repository or from a repository to a report. The first time that you use webElements, you must install the functions to the BusinessObjects Enterprise repository that will host your reports.

You must install the custom function suite from the repository to each report that will utilize webElements.

### Adding webElements to a Repository

In order to correctly display webElements in a report, you must install the most recent webElements functions to BusinessObjects Enterprise.

1. Open the webElements\_433.rpt file included with the webElements installation files.
2. From the **Report** menu, click **Formula Workshop**
3. Expand the **Report Custom Functions > webElements 4.33 > Admin** nodes.
4. Right click on the WEInstaller function and select **Add to Repository**.
5. When prompted to add the additional custom functions to the repository, click **Yes**.

**Note:** The installation process may take several minutes.

### Adding webElements to a New Report

To use webElements in a report, you must add the webElements custom functions suite to the report from a BusinessObjects Enterprise repository. You must repeat this step for each report that uses the webElements custom functions.

1. Open a report in Crystal Reports 2020.
2. From the **Report** menu select **Formula Workshop**.
3. Expand the **Repository Custom Functions> Repository >webElements 4.33 > Admin** nodes.
4. Right click on the WEInstaller function and select **Add to Report**.
5. When prompted to add the additional custom functions to the report, click **Yes**.

The webElements suite is now ready to use in your report. The suite must be added to any report or sub-report in which webElements are used.

## Removing webElements

To uninstall webElements from either a BusinessObjects Enterprise repository or a report file you must manually delete each function.

1. Open a report in Crystal Reports 2020 (later).
2. From the **Report** menu select **Formula Workshop**
  - Expand the **Repository Custom Functions** node to uninstall webElements from a BusinessObjects Enterprise repository
  - Expand the **Report Custom Functions** node to uninstall webElements from a report
3. Expand the **webElements 4.33** node
4. For each webElements function, select the function and press **Delete**

## Requirements for all Crystal Reports

Every report using webElements must contain the following:

1. A formula containing a WEBuilder function if any functions from the **webElements 4.33 > InputsAndControls** are used on a report.
2. A formula containing a submit button or link (found in the **webElements 4.33 > ButtonsAndLinks** node) is required if you wish to interact with the current report or to target another report
3. An input or control to interact with a current report target document (found in the Custom Functions in the **webElements 4.33 > InputsAndControls** node of the webElements\_433.rpt)
4. A WEPlatform function set to the version of BusinessObjects Enterprise that you are using. After installing the webElements 4.33 suite open the WEPlatform function from the **webElements 4.33 > Admin** node on your report. By default this is set for SAP BI / BusinessObjects 4.3

### Report sections containing webElements are less than one page in length only:

All webElements should be in a section that does not extend past one page. The report itself can be more than one page but webElements will not function properly when they are in a section that spans more than one page. This is due to a limitation in html and working with multi-page html documents that the viewer creates.

Note that the section that webElements are in can be repeated on more than one page as long as the section's length is less than one page...i.e. webElements can be in a Page Header section or in a repeated Group Header section that repeats on each page.

## Report Creation with webElements

You can copy formulae from the webElements\_428 report for easier report creation.

There are formulae on the webElements\_428 report that contain default syntax for each relevant webElements control. You can copy a formula that contains default syntax for any function to your new report. Afterwards, edit the function parameters to the desired values.

Use this report only in the Crystal Reports designer as it is not formatted for viewing in Launchpad. Ensure that you have installed webElements and have read the Requirements for all Crystal Reports section of the webElements User Guide.

To start building reports using webElements quickly:

1. Open the webElements\_433.rpt in the Crystal Reports designer
2. Find the type of control that you want to use on your report
3. Right click on the formula and choose Copy
4. Go to the report that you are developing and Paste the formula on to your report
5. Edit the formula to use your desired values and settings
6. Repeat steps 2 to 5 for any required controls.
7. Copy the WEBUILDER formula to a section below all of the other webElements formulae
8. Edit the WEBUILDER formula to include the formulae that you are using in your report
9. Save your report to your Enterprise environment and open your report in Launchpad to test
10. Consult the User Guide and sample reports for further help and more information

## How To Example

This section includes several steps that will walk you through the creation of a report that uses the webElements custom function suite.

### Creating a report that includes webElements

Before you can make use of any of the webElements functions, you must create a report, add the webElements functions, and save it to BusinessObjects Enterprise. This procedure is a pre-requisite for all other example.

1. Create a blank Crystal report using Crystal Reports 2020 (or later).
2. From the **Report** menu, select **Formula Workshop**.
3. Connect to the repository to which you have installed the webElements custom function suite.
  - Expand the **Repository Custom Functions** node.
  - Expand the node corresponding to your BusinessObjects enterprise server.
  - Enter your connection information and click **OK**.
4. Add the webElements functions to your report.
  - Expand the **webElements 4.33 > Admin** nodes.
  - Right-click on **WEInstaller**.
  - Click **Add to Report**.
  - When prompted to add multiple formulae to the report, click **OK**.
5. Close the **Formula Workshop**
6. Save the report to your BusinessObjects Enterprise repository as webelementsreport without the .rpt extension.

### Add a select menu to a report

This example demonstrates how to add a select menu control to a Crystal report using the webElements custom function suite.

**Pre-requisites** Before you can add a select menu to a report, you must first add the webElements custom functions to the report and save it to BusinessObjects Enterprise. See the section entitled "Installing webElements" for more information.

7. From the **Report** menu, select **Formula Workshop**.
8. Right click on the **Formula Fields** node and select **New**. When prompted, name the new formula 'selectmenu' and click **OK**.
9. In the **Formula Editor**, enter the following code.

```
stringvar ElementName:= "Select1";  
stringvar ElementValues:= "1|2|3|4";  
stringvar ElementDisplays:= "One|Two|Three|Four";  
stringvar ElementDefault:= "1";
```



```
WESelect (ElementName, ElementValues, ElementDisplays, ElementDefault, "")
```

10. Click **Save**.

11. Right click on the **Formula Fields** node and select **New**.

When prompted, name the new formula 'builder'. This formula will create all of the JavaScript variables needed for the select menu.

12. In the **Formula Editor**, create a new stringvar called allElements, and add the select function.

```
stringvar allElements:= {@selectmenu};
```

13. Below the allElements stringvar, enter the WEBUILDER function, and pass it the allElements stringvar, and set the debug level to '2'. By doing this, you will have a pop-up window that displays the OpenDocument URL created by your webElements formula.

```
WEBUILDER(allements, 2)
```

14. Click **Save and Close**.

15. In the **Field Explorer**, expand the **Formula Fields** node.

16. Drag and drop the calc formula from the Field Explorer to the report.

17. Drag and drop the builder formula from the Field Explorer to the report ensuring that the builder formula is below the selectmenu formula.

18. Save the report to your BusinessObjects Enterprise repository.

When you view the report in Launchpad, you should see a select menu control.

## Pass a prompt to a webElements control

This sample demonstrates how pass a report prompt value to a webElements control.

**Pre-requisites** You must first add the webElements custom functions to the report and save it to BusinessObjects Enterprise. See the section entitled "Creating a report that includes webElements" for more information. This sample assumes that you have completed the steps outlined in 'Add a select menu to a report', above.

1. In the **Field Explorer**, right-click on the **Parameter Fields** node and click **New**.

- Enter 'Select1' in the **Name** field. Note that the parameter name matches the ElementName that you gave to the select menu.
- Click **OK**.

2. Save the report to your BusinessObjects Enterprise repository.

When you view the report in Launchpad, you should be prompted for an Integer value. The Integer entered at the prompt screen will be used as the default number for the calculator control.

## Add a submit button to a report

This sample demonstrates how to add a submit button to a report and have the report update upon pressing the submit button.

**Pre-requisites** You must first add the webElements custom functions to the report and save it to BusinessObjects Enterprise. See the section entitled “Creating a report that includes webElements” for more information. This sample assumes that you have completed the steps outlined in ‘Pass a parameter to a webElements control’, above.

1. From the **Report** menu, select **Formula Workshop**.
2. Right click on the **Formula Fields** node and select **New**. When prompted, name the new formula ‘submit’ and click **OK**.
3. In the **Formula Editor**, enter the following code.

```
stringvar path:= WETargetPath ("rpt", "Name", "webelementsreport", "");  
WESubmitButton ("update", path, "")
```

4. Click **Save**.
5. Left click on the builder formula in the **Formula Fields** node.
6. In the **Formula Editor**, change the builder formula to the following code.

```
stringvar allElements:= {@selectmenu} + {@submit};  
WEBBuilder(allElements, 2)
```

7. Click **Save and Close**.
8. In the **Field Explorer**, expand the **Formula Fields** node.
9. Drag and drop the submit formula from the Field Explorer to the report ensuring that the submit formula is above the builder formula.
10. Save the report to your BusinessObjects Enterprise repository.

When you view the report in Launchpad, you should see a submit button. Pressing the submit button should repost the report with the current select menu value.

## webElements Function Reference

This section outlines each of the functions in the webElements custom function suite and its required parameters. Click on a function name to go to the detailed information for that function.

**Note:** see the formulas on the webElements433 report for an easier way to understand the functions and their syntax.

### Quick Reference

#### Main Elements

[WEBuilder](#)

The WEBuilder function generates the HTML required to run the other webElements functions. Required for most reports.

[WETargetPath](#)

Defines the target URL for report actions. Required for most reports.

#### Admin

[WEInstaller](#)

Used to install the webElements suite to a repository or to a report.

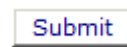
[WEPlatform](#)

Used to specify which version of the BusinessObjects Enterprise platform you are using.

#### Buttons and Links

[WESubmit](#)

Creates a button or link that submits all chosen input and control values to target URL's when pressed.



[WESubmitImage](#)

Uses a set of images that submit all chosen input and control values to target URL's when clicked.

[WESubmitLinkList](#)

Creates several links that submit all chosen input and control values to the target URL when clicked.

[Go to report 1](#)

[Go to report 2](#)

[Go to report 3](#)

[WESubmitLinkRadio](#)

Creates a radio button control that submits all chosen input and control values to target URL's when clicked.

[WESubmitLinkSelect](#)

Creates a drop-down list that submits all chosen

input and control values to the target URL when a selection is made.



Formatting

[WEBreak](#)

Adds one or more line breaks.

[WECollapsibleSection](#)

Creates a collapsible section in which text and controls can be displayed.

[WEFlyoutSections](#)

Creates a set of fly-out sections in which text and controls can be displayed.



[WEFont](#)

Used to format the fonts of most inputs and controls.

[WEFontEXT](#)

Used to format the fonts of most inputs and controls.

[WEIFrame](#)

Creates an IFrame within a report.

[WEImage](#)

Adds an image to a report.

[WEMarquee](#)

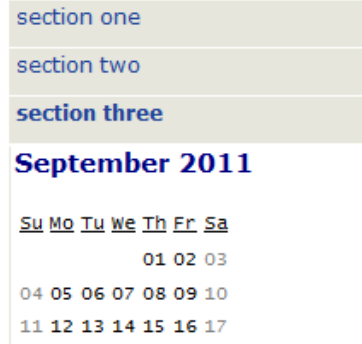
Creates a region of the report where content scrolls from right to left.

[WEMarqueeEXT](#)

Creates a region of the report where content scrolls from right to left.

[WEPulldownSections](#)

Creates a set of pull-down sections in which text and controls can be displayed.



[WESpace](#)

Adds a non-breaking space.

[WETextObject](#)

Creates a formatted block of text.

[WETextObjectExt](#)

Creates a formatted block of text.

### Form Controls

[WEControlDisplaySelect](#)

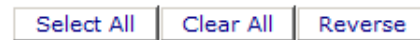
Creates a drop-down list that will hide or show other controls.

[WEResetButton](#)

Resets controls referenced in the formula referencing WEBUILDER to their default values.

[WESelectAllClearAllReverseButtons](#)

Creates a set of three buttons that will set, clear, or reverse all values of a control.



[WESelectAllClearAllReverseLink](#)

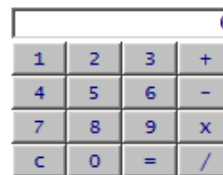
Creates a set of three hyperlinks that will set, clear, or reverse all values of a control.

Select All Clear All Reverse

### Inputs and Controls

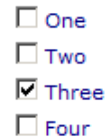
[WECalculator](#)

Adds a calculator to the report.



[WECheckBox](#)

Creates a check box control that allows the user to choose one or more values from a collection of checkboxes.



☐ One  
☐ Two  
☒ Three  
☐ Four

#### [WEComboBoxSelect](#)

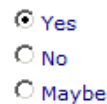
Creates a combination text box and select control that allows the user to search for values in the select using the text box.



Aceves, Alfredo  
Accardo, Jeremy  
Aceves, Alfredo  
Acosta, Manny

#### [WERadio](#)

Creates a radio button control that allows the user to choose only one value from a set of choices.



☒ Yes  
☐ No  
☐ Maybe

#### [WESelect](#)

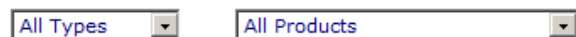
Creates a drop-down menu that allows the user to choose a single value from a set of choices.



Blue  
Blue  
Yellow  
Red  
Green  
Orange  
Purple

#### [WESelectCascade](#)

Creates a drop-down menu that is part of a cascading prompt set. This type of control allows the user to choose a single value from a set of choices.



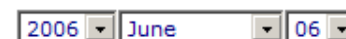
All Types All Products

#### [WESelectCascadeExt](#)

Creates a drop-down menu that is part of a cascading prompt set. This control allows for the value passed to the result URL to be different than the selection options.

#### [WESelectDate](#)

Creates a set of three drop-down menus to select a year, month, and day.



2006 June 06

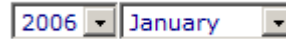
#### [WESelectDateRange](#)

Creates two sets of three drop-down menus to select a year, month, and day.

#### [WESelectDuo](#)

Creates two drop-down menus. When the page is submitted, the values from both drop-down menus will be concatenated

and passed as a single parameter to the URL.



#### [WESelectMulti](#)

Creates a list of items that allows a user to select one or more items from a list.



#### [WESelectMultiCascade](#)

Creates a drop-down menu that is part of a cascading prompt set.



#### [WESelectMultiCascadeExt](#)

Creates a drop-down menu that is part of a cascading prompt set. This control allows for the value passed to the result URL to be different than the selection options.

#### [WESelectTrio](#)

Creates three drop-down menus. When the page is submitted, the values from the three drop-down menus will be concatenated and passed as a single parameter to the URL.



#### [WESpin](#)

Creates a “spinner” type control that allows a user to increment and select a number.



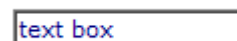
#### [WETextArea](#)

Creates a multi-line text area input control.



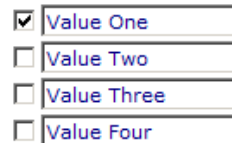
#### [WETextBox](#)

Creates a single line text box control.



#### [WETextBoxAndCheckBox](#)

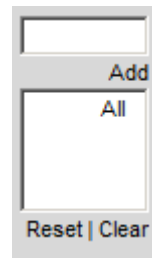
Creates a text input that is bound to a checkbox control. The value of both the text box and checkbox are passed to the result URL.



A screenshot of a web control with four rows. Each row has a checkbox on the left and a text box on the right. The first row has a checked checkbox and the text 'Value One'. The second row has an unchecked checkbox and the text 'Value Two'. The third row has an unchecked checkbox and the text 'Value Three'. The fourth row has an unchecked checkbox and the text 'Value Four'.

#### [WETextBoxMulti](#)


Creates a multiple value text box control.



A screenshot of a web control. It features a small text box at the top containing the word 'All'. Below this text box is a button labeled 'Add'. At the bottom of the control are two buttons labeled 'Reset' and 'Clear'.

#### [WETreePicker](#)

Creates a tree picker control where chosen tree menu values are displayed in a text area.



A screenshot of a web control. It features a text area containing the text 'Product Line'. Below the text area are two buttons labeled 'Reset' and 'Clear'.

- Product
  - > Product Line
  - > Product Category
  - > Product Name
- Geography
  - > Region

#### [WETreePickerExt](#)

Creates a tree picker control where chosen tree menu values are displayed in a text area. This control allows for the value passed to the result URL to be different than the selection options.

### Menus

#### [WETreeMenu](#)

Adds a tree menu to a report.

### Other

#### [ArrayPositionFinder](#)

Finds the position of a specific value in a String Array.

#### [WEAutoRefresh](#)

Refreshes a report automatically.



**Target Paths**[WETargetPathExt](#)

Defines the target URL for submit actions and defines the target window properties.

## Main Elements

**Note:** see the formulas on the `webElements433.rpt` for an easier way to understand the functions and their syntax.

Requirement: One formula containing the `WEBuilder` function is required to generate the web objects in the Crystal Report.

### WEBuilder(FormElements, ProductionMode)

The `WEBuilder` function generates the HTML required to run the other webElements functions. A single `WEBuilder` function must appear in an unsuppressed section of the report.

#### Parameters:

**FormElements** is a String list of all webElements in the report.

**ProductionMode** is used by the report developer for debugging the URL output. `ProductionMode` has three possible values:

- If `ProductionMode` is set to '1' the result URL will appear in a pop-up alert. The target URL will not be run.
- If `ProductionMode` is set to '2' the result URL will appear in a pop-up alert and the target URL will run.
- If `ProductionMode` is set to '3' the target URL will run but no pop-up alert will appear.

Set `ProductionMode` to either '1' or '2' during when designing and debugging a report. Set `ProductionMode` to '3' when placing the report into a production environment.

### WESubmit (ElementName, Type, LinkText, Path, ElementFont)

The `WESubmit` function creates a link or button that submits all selected input and control values to the target URL when clicked.

The `WESubmit` function can be placed in a different formula, section or group from the webElements controls it affects.

#### Parameters:

**ElementName** is a unique name assigned to the control. This name does not have to match a prompt / parameter name.

**Type** determines what kind of submit control is used. This parameter is either "Link" or "Button".

**LinkText** is the text that will appear on the link or button.

**Path** is a URL to the target report. For example:

```
WETargetPath ("rpt", "Name", "Sales Report", "weWindow=New")
```

Using `weWindow=New` in the last parameter of `WETargetPath` will open the report in a new window each time.

Using `weWindow=yourname` in the last parameter of `WETargetPath` will open the report in a named window.

**ElementFont** is used to format the text on the link or button. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style.

### WETargetPath(DocType, IDType, ID, OtherParams)

The WETargetPath function creates a target URL. This target URL is used to pass parameters to other reports in the system. WETargetPath utilizes the OpenDocument function to link to other reports.

#### Parameters:

**DocType** is a String value that represents the type of document being linked to. DocType can be set to one of the following values.

- “rpt” for a Crystal report

**IDType** represents the type of the ID passed in the ID parameter. Set IDType to “CUID” if you are linking to an object in the repository by its ID. Set IDType to “Name” if you are linking to an object in the repository by its name.

**ID** is either the CUID or the Name of the object being linked to.

The type of value passed to the ID parameter depends upon the value passed to the IDType parameter.

**OtherParams** are any other parameters you wish to pass in the URL string that will not be passed by the webElements functions. For example, set OtherParams to &lsCountry=“USA” to set the country prompt of the report to USA.

Use the OtherParams parameter to pass control values to an IFrame, a named window / named viewer.

To pass control values to an IFrame, set OtherParams to “weiframe=” plus the IFrame name. This is useful if you wish to pass values to a Crystal Report document that is embedded into your main report with the WEIFrame function.

To pass control values to a new window each time, set OtherParams to “weWindow=New”.

To pass control values to a name window, set OtherParams to “weWindow=” plus the window name. This is useful if you wish to pass values to a Crystal report document that is in a named window. Using named windows will prevent a new window from being opened each time.

#### Remarks:

The WETargetPath function specifies the default location of the openDocument function for your installation of BusinessObjects Enterprise. The default location is selected as a result of the value specified in the WEPlatform function.

## Admin

This section includes functions in the Admin folder of the webElements function library. Administrative functions are used to install or validate webElements functions. These functions are not used in regular report design.

### WEInstaller

The WEInstaller function is used to install the webElements suite to a repository or to a report. See the section entitled Installing webElements for more information.

The first time that you use webElements, you must install the functions to the BusinessObjects Enterprise repository that will host your reports. You must then install the custom function suite from the repository to each report that contains webElements.

## Admin > Source

### WEPlatform

This function is used to specify which version of the BusinessObjects Enterprise platform you are using. By default, it is set to the BusinessObjects Enterprise 4.3.

### WEValidator

WEValidator is used to validate inputs and controls used in a report. Calls to WEValidator are automatically made by other functions. This function does not need to be modified or called directly in any report formulae.

## Buttons and Links

This section includes functions in the ButtonsAndLinks folder of the webElements function library. These functions will add buttons, hyperlinks, or hyperlinked images to your report that can then be used to submit a URL to a target window or IFrame.

### WESubmitMulti (ElementName, Type, LinkText, Paths, ElementFont)

The WESubmitMulti function creates a link or button that submits all selected input and control values to the target URL when clicked.

The WESubmitMulti function can be placed in a different formula, section or group from the webElements controls it affects.

#### Parameters:

**ElementName** is a unique name assigned to the control. This name does not have to match a prompt / parameter name.

**Type** determines what kind of submit control is used. This parameter is either "Link" or "Button".

**LinkText** is the text that will appear on the link or button.

**Paths** is a character separated list of the URL paths to the target reports. Use the "|" character to separate multiple elements. When the end user presses the submit control, all targets defined in the paths parameter will be opened. For example:

```
WETargetPath ("rpt", "Name", "Sales Report", "weWindow=New") + "|" + WETargetPath ("rpt", "Name", "Cost Report", "weWindow=namedwindow1")
```

Use the WETargetPathExt function if you wish to have a report open in a custom window where the size and browser toolbars are specified.

Using weWindow=New in the last parameter of WETargetPath will open the report in a new window each time.

Using weWindow=yourname in the last parameter of WETargetPath will open the report in a named window.

**ElementFont** is used to format the text on the link or button. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style.

### WESubmitImage(ElementName, ImageLocations, Paths, ToolTip, Width, Height)

The WESubmitImage function creates a hyperlinked image set that allows the user to submit all chosen input and control values to the target URL.

The WESubmitImage function does not have to be in the same formula as any of the controls or in the same area or section of the report where the controls are.

#### Parameters:

**ElementName** is a unique name assigned to the control. This name does not have to match a prompt / parameter name.

**ImageLocations** is a character separated list that contains the paths to the images (e.g. `http://server.com/path/imagename.gif`) or file paths to the images (e.g. `c:\folder\imagename.gif`). Use the "|" character to separate multiple image files.

Up to three (3) images can be used. The first image is the 'base' image, the second image is the 'hover over' image, and the third image is the 'on click' image.

**Paths** is a character separated list of the URL paths to the target reports. Use the "|" character to separate multiple elements. For example:

```
WETargetPath ("rpt", "Name", "Sales Report", "") + "|" + WETargetPath ("rpt", "Name", "Cost Report", "")
```

**Tooltip** is the text that will appear as a browser tooltip when a mouse cursor is hovered over the image.

**Width** is a String that represents the image width. Use an integer such as "66" to give a width in pixels, "2in" for a width in inches, or "2cm" for a width in centimeters. Leave the width blank "" to use the default width of the image.

**Height** is a String that represents the image height. Use an integer such as "66" to give a height in pixels, "2in" for a height in inches, or "2cm" for a height in centimeters.. Leave the height blank "" to use the default height of the image.

### WESubmitLinkList(LinkTexts, Paths, AlignVertically, Spacing, ElementFont)

The WESubmitLinkList function creates a list of hyperlinks that allows the user to choose a target URL and to submit all selected input and control values to that target URL.

The WESubmitButton element can be placed in a different formula, section or group from the webElements controls it affects.

#### Parameters:

**ElementName** is a unique name assigned to the control. This name does not have to match a prompt / parameter name.

**LinkTexts** is a character separated list of the text that will appear for the hyperlinks. Use the "|" character to separate multiple elements.

E.g. ">> Go to sales report|>> Go to cost report".

**Paths** is a character separated list of the URL paths to the target reports. Use the "|" character to separate multiple elements. For example:

```
WETargetPath ("rpt", "Name", "Sales Report", "") + "|" + WETargetPath ("rpt", "Name", "Cost Report", "")
```

**AlignVertically** is a Boolean value. Set AlignVertically to True to align the URLs vertically. Set AlignVertically to False to align the URLs in one horizontal line.

**Spacing** is a number value representing the number of line breaks between URLs if AlignVertically is set to True, or the number of spaces between URLs if AlignVertically is set to False.

**ElementFont** is used to format the text of the hyperlink. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style.

**Note:** This is a single value that will format all URLs generated by this function.

### WESubmitLinkRadio(ElementName, LinkTexts, Paths, ElementDefault, SubmitLinkLabel, AlignVertically, ElementFont, InvalidMessage)

The WESubmitLinkRadio function creates a radio button control. When an item is selected from the control, the page will submit all selected input and control values to the target URL.

The WESubmitLinkRadio element can be placed in a different formula, section or group from the webElements controls it affects.

#### Parameters:

**ElementName** is a unique name assigned to the control. This name does not have to match a prompt / parameter name.

E.g. "SLR1"

**LinkTexts** is a character separated list of the text that will be beside the radio buttons. Use the "|" character to separate multiple elements.

E.g. "Sales Report|Logistics Report".

**Paths** is a character separated list of the URL paths to the target reports. Use the "|" character to separate multiple elements. For example:

```
WETargetPath ("rpt", "Name", "Sales Report", "") + "|" + WETargetPath ("rpt", "Name", "Cost Report", "")
```

**ElementDefault** is used to preselect a radio control member.

E.g. WETargetPath ("rpt", "Name", "Sales Report", "")

**SubmitLinkLabel** is an optional link that will appear below the radio button control.

If SubmitLinkLabel is left blank, "", then the URL will be generated whenever a radio button is selected. If SubmitLinkLabel is not blank, e.g. ">> Submit", then the URL will only be generated once the link below the radio control is clicked.

The font for SubmitLinkLabel is, by default, the same font used by the radio control text displays. The SubmitLinkLabel font can be changed by editing the syntax of the WESubmitLinkRadio function. Further instructions are in the function syntax.

**AlignVertically** is a Boolean value. Setting AlignVertically to True aligns the radio buttons and display values vertically. Setting this value to False or leaving the parameter empty places the radio buttons and display values in one horizontal line.

**ElementFont** is used to format the text of the display elements. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style. **Note:** This is a single value that will format all URLs generated by this function.

**InvalidMessage** is the message that will be displayed in an alert pop-up if no radio buttons are selected at the time of submission. Leaving this parameter blank, "", will not activate validation for the control.

### WESubmitLinkSelect(ElementTitle, LinkText, Path, ElementFont)

The WESubmitLinkSelect function creates a drop-down list. When an item is selected from the list, the page will submit all selected input and control values to the target URL.

The WESubmitButton element can be placed in a different formula, section or group from the webElements controls it affects.

#### Parameters:

**ElementName** is a unique name assigned to the control. This name does not have to match a prompt / parameter name.

**ElementTitle** will appear as the first item in the drop-down list. It is used only as a display and will not create a hyperlink.

E.g. "Select target report"

**LinkText** is a character separated list of the text that will appear for the hyperlinks. Use the "|" character to separate multiple elements.

E.g. ">> Go to sales report|>> Go to cost report".

**Path** is a character separated list of the URL paths to the target reports. Use the "|" character to separate multiple elements. For example:

```
WETargetPath ("rpt", "Name", "Sales Report", "") + "|" + WETargetPath ("rpt", "Name", "Cost Report", "")
```

**ElementFont** is used to format the text of the hyperlink. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style.

**Note:** This is a single value that will format all URLs generated by this function.



## Formatting

This section includes functions in the Formatting folder of the webElements function library. These functions can be used to add images to your report, change the font of text elements, or manually insert spacing.

### WEBreak(N)

The WEBreak function is used to create one or more line breaks between elements if you have more than one element in a formula.

#### Parameters:

**N** is an integer representing the number of line breaks to place between elements.

#### Example:

The following is an example formula containing the WEBreak function. This formula creates a checkbox control followed by, two line breaks, and a drop-down list.

```
WECheckBox("Country", "Canada|USA|Mexico", "Canada|USA|Mexico", "Canada", True, "",
"Checked", "Please select at least one country.") +
WEBreak(2) +
WESelect("Value", "1|2|3|4|5|6", "One|Two|Three|Four|Five|Six", "One", "")
```

### WECollapsibleSection(ElementName, SectionElements, OpenText, CloseText, ElementFont, SectionWidth, SectionHeight)

The WECollapsibleSection function creates a section that can be opened and closed on a report. These sections can contain webElements text objects, inputs, and controls.

The WECollapsibleSection function has syntax that can be further edited to match the desired look and feel. Open this function in the Formula Workshop of Crystal Reports for further information. The background, section margins, border, and open / close icons and fonts can be modified within the function syntax.

#### Parameters:

**ElementName** is a unique name that the report developer assigns to the section.

**SectionElements** is a String list of all of the webElements in the section.

**OpenText** is a String that will be displayed when the section is closed (e.g. "Open prompt section").

**CloseText** is a String that will be displayed when the section is open (e.g. "Close this section now").

**ElementFont** formats both the OpenText and CloseText displays. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style.

**SectionWidth** is a String that can be in inches (e.g. "3in") centimeters (e.g. "6cm"), or pixels (e.g. "1600pi").

**SectionHeight** is a String that can be in inches (e.g. "3in") centimeters (e.g. "6cm"), or pixels (e.g. "1600pi").

#### Example:

The following is an example formula containing the WECollapsibleSection function.

```
stringvar sectionfonts := WEFont ("Verdana", 8, "Blue", "Center", True, False, "Underline",
    "Navy");
stringvar sectionelements:= {@selectmenu} + {@submitbutton};
stringvar collapsiblesection1:= WECollapsibleSection ("section1", sectionelements, "View
    Controls", "Hide Section", "sectionfonts", "3in", "4in")
```

### WEFlyoutSection(ElementName, ElementHeight, ContentWidth, ElementHeaders, ElementContent)

The WEFlyoutSection function creates a set of fly-out sections that can be opened and closed on a report. These sections can contain webElements text objects, inputs, and controls.

The WEFlyoutSection function has syntax that can be further edited to match the desired look and feel. Open this function in the Formula Workshop of Crystal Reports for further information. The background, section margins, border, and fonts can be modified within the function syntax.

For best results, also ensure that plain or formatted text that you wish to use in a WEFlyoutSection is inside a WETextObject or a WETextObjectExt function.

#### Parameters:

**ElementName** is a unique name that the report developer assigns to the set of sections.

**ElementHeight** is a String that can be in inches (e.g. "3in") centimeters (e.g. "6cm"), or pixels (e.g. "1600pi").

**ContentWidth** is a String that can be in inches (e.g. "3in") centimeters (e.g. "6cm"), or pixels (e.g. "1600pi"). This sets the width of each individual section.

**ElementHeaders** is a String Array of the text that will be displayed in the individual section headers.

**ElementContent** is a String Array of the content that will be displayed in each section.

Ensure that the number of items in the ElementHeaders array matches the number of items in the ElementContent array.

#### Example:

The following is an example formula containing the WEFlyoutSection function.

```
stringvar fsheight:= "6cm";
stringvar fswidth:= "4cm";
stringvar array fsheaders:= ["section one","section two","section three","section four"];
stringvar array fscontent:= [{@text1},{@iframe},{@calendar},{@text3}];
WEFlyoutSections ("fs1", fsheight, fswidth, fsheaders, fscontent)
```

### WEFont (FType, FSize, FColour, FAlignment, FBold, FItalic, FDecoration, FBackground)

The WEFont function is used to format the fonts of any input or control elements. It is used with any element that has a Font parameter in the function.

**Parameters:**

**FontType** is a string value that defines the font to use. The default value is Verdana.

**FontSize** is a number that defines the size of the font in points. The default value is 9 pt.

**FontColour** defines colour of the text. This can be a string such as "Green" or a hex value such as "B51626". The default value is black.

**FontAlignment** will change the alignment of the text. Choose one of the following values: "Left", "Right", "Justify" or "Center". The default value is left aligned.

**FontBold** is a Boolean value. Set to True to make the font bold face. The default value is False.

**FontItalic** is a Boolean value. Set to True to make the font italicized. The default value is False.

**FDecoration** is a string variable used for the font decoration.

Choose one of the following values: "", "underline", "overline", or "line-through". The default value is "None".

**FBackground** defines the background colour of the text. This can be a string such as "Green" or a hex value such as "B51626".

**Example:**

The following is an example formula containing the WEFont function.

```
stringvar checkboxfont := WEFont ("Verdana", 8, "Blue", "Center", True, False, "Underline",
    "Navy");
stringvar checkboxcontrol := WECheckBox("Country", "Canada|USA|Mexico",
    "Canada|USA|Mexico", "Canada", True, checkboxfont, "Checked", "Please select at least
    one country.")
```

### **WEFontEXT (FType, FSize, FColour, FAlignment, FBold, FItalic, FDecoration, FBackground)**

The WEFontEXT function is used to format the fonts of any input or control elements. It is used with any element that has a Font parameter in the function.

Although WEFontEXT shares identical parameters to WEFont its syntax can be modified in the Formula Workshop of Crystal Reports to format more properties.

These properties are activated when the user's mouse hovers over an object, when the user clicks on an object, or when the user's mouse stops hovering over an object.

### **WEImage(Path, Width, Height)**

The WEImage function is used to render a picture in the report viewer.

This function does not require a WEBUILDER function on the report.

**Parameters:**

**Path** is a String that represents the virtual path to the image (e.g. <http://server.com/path/imagename.gif>) or file path to the image (e.g. <c:\folder\imagename.gif>).

**Width** is a String that represents the image width. Use an integer such as “66” to give a width in pixels, “2in” for a width in inches, “2cm” for a width in centimeters, or “10%” for a percentage of the original width. Leave the width blank “” to use the default width of the image.

**Height** is a String that represents the image height. Use an integer such as “66” to give a height in pixels, “2in” for a height in inches, “2cm” for a height in centimeters, or “10%” for a percentage of the original height. Leave the height blank “” to use the default height of the image.

### WEMarquee(ElementName, ElementDisplay, ElementWidth, ScrollSpeed, ScrollAmount)

Note: This function may not work in all browsers. For best results use Microsoft Edge.

The WEMarquee function is used to scroll text and links within a Crystal Report. This function is useful for displaying selected prompt values in a limited amount of space.

By default, content scrolls from right to left. The scroll direction can be edited within the function itself. Open the WEMarquee function in the **Report** menu, **Formula Workshop** for more information.

This function does not require a WEBUILDER function on the report.

#### Parameters:

**ElementName** is a text value representing a unique name assigned to each WEMarquee function on a report. This name is not associated with a prompt as the function is used only to display information.

**ElementDisplay** is a text value representing the content that will display in the report.

**ElementWidth** is a String that represents the scroll width. Use an integer such as “66” to give a width in pixels, “2in” for a width in inches, or “2cm” for a width in centimeters.

**ScrollSpeed** is an integer value that determines the speed at which the content moves. 1 is the slowest scroll speed and 100 is the fastest scroll speed.

**ScrollAmount** is an integer value that determines the length of content that is moved at each increment. 1 is the smallest length and 100 is the largest length of content that is moved.

### WEMarqueeExt(ElementName, ElementDisplay, ElementWidth, ScrollSpeed, ScrollAmount, ScrollDirection, ControlPositions, ControlToolTips)

Note: This function may not work in all browsers. For best results use Microsoft Edge.

The WEMarqueeExt function is used to scroll text and links and other content within a Crystal Report.

The WEMarqueeExt function syntax can be edited to create a desired look and feel for your tab menu. Control icons, fonts, tooltips, etc. can be customized within the function syntax. Open the WEMarqueeExt function in the **Report** menu, **Formula Workshop** for more information.

This function does not require a WEBUILDER function on the report.

#### Parameters:

**ElementName** is a text value representing a unique name assigned to each WEMarquee function on a report. This name is not associated with a prompt as the function is used only to display information.

**ElementDisplay** is a text value representing the content that will display in the report.

**ElementWidth** is a String that represents the marquee width. Use an integer such as "66" to give a width in pixels, "2in" for a width in inches, or "2cm" for a width in centimeters.

**ScrollSpeed** is an integer value that determines the speed at which the content moves. 1 is the slowest scroll speed and 100 is the fastest scroll speed.

**ScrollAmount** is an integer value that determines the length of content that is moved at each increment. 1 is the smallest length and 100 is the largest length of content that is moved.

**ScrollDirection** is a text value that determines the direction that the content should scroll. Values include "left", "right", "up", or "down".

**ControlPositions** is a text value that determines the position of the start, stop, and reverse controls for the marquee. Values include "bottom", "top", "left", or "right".

**ControlTooltips** is a character separated list of the text that will appear for the start, stop, and reverse controls. Use the "|" character to separate multiple elements. This parameter can be left blank and the defaults of "Left" or "Up", "Stop", "Right" or "Down" will be used depending on whether the marquee scrolls vertically or horizontally.

#### Example:

The following is an example formula containing the WEMarqueeExt function.

```
stringvar mcontent:= {@image1} + {@image2} + {@image3};
stringvar mwidth:= "6cm";
numbervar mscrollspeed:= 90;
numbervar mscrollamount:= 2;
stringvar mscrolldirection:= "left";
stringvar mcontrolpositions:= "bottom";
WEMarqueeExt ("m1", mcontent, mwidth, mscrollspeed, mscrollamount, mscrolldirection,
mcontrolpositions, "")
```

### WEPulldownSection(ElementName, ElementHeight, ContentWidth, ElementHeaders, ElementContent)

The WEPulldownSection function creates a set of fly-out sections that can be opened and closed on a report. These sections can contain webElements text objects, inputs, and controls.

The WEPulldownSection function has syntax that can be further edited to match the desired look and feel. Open this function in the Formula Workshop of Crystal Reports for further information. The background, section margins, border, and fonts can be modified within the function syntax.

For best results, also ensure that plain or formatted text that you wish to use in a WEPulldownSection is inside a WETextObject or a WETextObjectExt function.

#### Parameters:

**ElementName** is a unique name that the report developer assigns to the set of sections.

**ElementHeight** is a String that can be in inches (e.g. "3in") centimeters (e.g. "6cm"), or pixels (e.g. "1600pi").

**ContentWidth** is a String that can be in inches (e.g. "3in") centimeters (e.g. "6cm"), or pixels (e.g. "1600pi"). This sets the width of each individual section.

**ElementHeaders** is a String Array of the text that will be displayed in the individual section headers.

**ElementContent** is a String Array of the content that will be displayed in each section.

Ensure that the number of items in the ElementHeaders array matches the number of items in the ElementContent array.

**Example:**

The following is an example formula containing the WEPulldownSection function.

```
stringvar psheight:= "6cm";
stringvar pswidth:= "4cm";
stringvar array psheaders:= ["section one", "section two", "section three", "section four"];
stringvar array pscontent:= [{"@text1"},{@iframe},{@calendar},{@text3}];
WEPulldownSection ("ps1", psheight, pswidth, psheaders, pscontent)
```

### **WEScroll(ElementName, ElementDisplay, ElementHeight, ElementWidth, ScrollLength, ScrollSpeed)**

The WEScroll function is used to scroll text and pictures within a Crystal Report. Content scrolls from top to bottom. This function is useful for displaying selected prompt values in a limited amount of space.

This function does not require a WEBUILDER function on the report.

**Parameters:**

**ElementName** is a text value representing a unique name assigned to each WEScroll function on a report. This name is not associated with a prompt as the function is used only to display information.

**ElementDisplay** is a text value representing the content that will display in the report.

**ElementHeight** is an integer representing the height, in pixels, of the scroll window in the report.

**ElementWidth** is an integer representing the width, in pixels, of the scroll window in the report.

**ScrollLength** is an integer representing the overall length, in pixels, of the scroll inside the window. It is recommended that you experiment with this value until all content scrolls from end to end.

**ScrollSpeed** is an integer value that determines the speed at which the content moves. 1 is the slowest scroll speed and 100 is the fastest scroll speed.

## WESpace(N)

The WESpace function is used to create a space between elements if you have more than one element in a formula.

### Parameters:

**N** is an integer representing the number of spaces to place between elements.

Note: WETextObject may not have text wrap in some browsers, such as Safari. Try using WETextObjectExt instead.

WETextObject can be used as an inline object and WETextObjectExt cannot be used inline. This affects the output when these functions are used in the same formula as other webElements functions.

As a result, WETextObject is good for short text, and WETextObjectExt is good for longer text.

## WETextObject(ElementDisplay, ElementFont)

The WETextObject function is used to create a formatted block of text in the report viewer. A WETextObject element is required for the WEScroll element. This element can also be used to insert formatted text blocks between WebElement controls when all controls are written in a single formula.

This function does not require a WEBUILDER function on the report.

### Parameters:

**ElementDisplay** is the text that will be shown in the report viewer.

**ElementFont** is used to format the text displayed by the WETextObject element. The report developer can use the WEFONT function in the Formatting functions to easily define a formatted font style.

## WETextObjectExt(ElementDisplay, ElementFont, ElementHeight, ElementWidth, ElementScroll)

The WETextObjectExt function is used to create a formatted block of text in the report viewer. A WETextObjectExt can be used in the WEScroll or WEMarquee elements. This element can also be used to insert formatted text blocks between WebElement controls when all controls are written in a single formula.

This function does not require a WEBUILDER function on the report.

### Parameters:

**ElementDisplay** is the text that will be shown in the report viewer.

**ElementFont** is used to format the text displayed by the WETextObject element. The report developer can use the WEFONT function in the Formatting functions to easily define a formatted font style.

**ElementHeight** is a String that represents the text object height. Use an integer such as "66" to give a width in pixels, "2in" for a width in inches, or "2cm" for a width in centimeters.

**ElementWidth** is a String that represents the text object width. Use an integer such as "66" to give a width in pixels, "2in" for a width in inches, or "2cm" for a width in centimeters.

**ElementScroll** is a Boolean that dictates whether scrollbars will appear in the text object should its content exceed its size. Setting this to false will clip the content that exceeds the height and width of the text object.

## Form Controls

This section includes functions in the Form Controls folder of the webElements function library. These functions can be used to reset controls or reset all form controls.

### WEControlDisplaySelect (ElementName, ElementDisplays, ElementControls, ElementDefault, Prompt, ElementFont)

The WEControlDisplaySelect function creates a drop-down list. When an item is select from the list, all controls that are associated with that item will be displayed. All other controls will be hidden.

#### Parameters:

**ElementName** is a unique name assigned to the control. This name must match a prompt / parameter name only if the chosen value is to be added to the OpenDocument URL.

**ElementDisplays** is a list of the menu items that will be displayed for the control sets. Use the "|" character to separate multiple elements.

Use "All" as an option for showing all controls. Use "None" as an option for hiding all controls.

**ElementControls** is a list of the actual controls that are associated with the ElementDisplays. Use the "|" character to separate multiple elements.

Use "All" as an option for showing all controls. Use "None" as an option for hiding all controls.

**ElementDefault** will appear as the first item in the drop down list. Any controls that are associated with the default value will be shown.

The ElementDefault must be an item from the ElementDisplays list.

**Prompt** is a Boolean variable that allows the chosen value of the control to be added to the OpenDocument URL.

**ElementFont** is used to format the text on the control. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style.

#### Example:

The following is an example formula containing the WEControlDisplaySelect function.

```
stringvar ElementName:= "cs1";

stringvar ElementDisplays:=
"All|None|Tree|Select Menus|Select Menus|Calendar";

stringvar ElementControls:=
"All|None|TreePicker1|Select1|Select2|Radio1|Calendar1";

stringvar ElementDefault:= "Select Menus";

booleanvar AddThisValueToURL:= true;
// this variable is used for the "Prompt" value of the function

WEControlDisplaySelect (ElementName, ElementDisplays, ElementControls, ElementDefault,
AddThisValueToURL, "")
```

### WEIframe(ElementName, Path, Width, Height, Scroll)

The WEIframe function is used to render a web page within the report viewer. The frame can be used to display another portion of your web portal, another report, etc.



The WEIframe function syntax can be edited to allow for a transparent background. Open the WEIframe function in the **Report** menu, **Formula Workshop** for more information.

**Parameters:**

**ElementName** is a unique name that the report developer assigns to the IFrame. This is important if a report is being developed with a report viewer IFrame that is targeted with controls on the report.

**Path** is a String that represents the virtual path to the frame display.

**Width** is a Number that represents the frame width in pixels.

**Height** is a Number that represents the frame height in pixels.

**Scroll** is a String value used to turn scrolling within the IFrame on or off. Set to "Yes" or "On" to enable scrolling, set to "No" or "Off" to disable scrolling. This will not overwrite any scroll code in the page being displayed within the frame.

**Example:**

The following is an example formula containing the WEIframe function.

```
WEIframe ("Iframe1", "https://www.crystalreports.com/", "400", "600", "No")
```

### WEResetButton(ButtonText, ButtonFont)

The WEResetButton function creates a reset button that will set all webElements inputs and controls to their default values when clicked. The WEResetButton element can be placed in a different formula, section or group from the webElements controls it affects.

**Parameters:**

**ButtonText** is the text that will appear on the button. For example, "Reset All Values".

**ButtonFont** is used to format the text on the button and the colour of the button. Use the WEFont function to define a specific font or style.

### WESelectAllClearAllReverseButtons(ElementName, ButtonText, ButtonFont)

The WESelectAllClearAllReverseButtons function creates three buttons in the report.

- A button to select all values of a specific control
- A button to clear all selected values of a specific control
- A button to reverse the selection of a specific control

The WESelectAllClearAllReverseButtons function must be used with the WESelectMulti, WECheckBox, or the WETextBoxAndCheckBox functions. The WESelectAllClearAllReverseButtons function can be placed in a different formula, section or group from other webElements controls.

**Parameters:**

**ElementName** is the same ElementName as the WESelectMulti or WECheckBox that is associated with the buttons. For example, if your report has a checkbox for a {?Country} prompt, then both the WECheckBox and the WESelectAllClearAllReverseButtons will have an ElementName of "Country".

**ButtonText** is a character separated list of the text that will be displayed on the buttons. If left blank, "", the buttons will default to "Select All", "Clear All", and "Reverse".

**ButtonFont** is used to format the text on the buttons and the colour of the buttons. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style.

### WESelectAllClearAllReverseLinks(ElementName, LinkText, ElementFont, AlignVertically)

The WESelectAllClearAllReverseLinks function creates three hyperlinks in the report.

- A link to select all values of a specific control
- A link to clear all selected values of a specific control
- A link to reverse the selection of a specific control

The WESelectAllClearAllReverseLinks function must be used with the WESelectMulti, WECheckBox, or the WETextBoxAndCheckBox functions. The WESelectAllClearAllReverseLinks function can be placed in a different formula, section or group from other webElements controls.

#### Parameters:

**ElementName** is the same ElementName as the WESelectMulti or WECheckBox that is associated with the links. For example, if your report has a checkbox for a {?Country} prompt, then both the WECheckBox and the WESelectAllClearAllReverseLinks will have an ElementName of "Country".

**LinkText** is a character separated list of the text that will be displayed for the hyperlinks. If left blank, "", the links will default to "Select All", "Clear All", and "Reverse".

**ElementFont** is used to format the text of the hyperlinks. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style.

**AlignVertically** is a Boolean value. Set AlignVertically to True to align the hyperlinks vertically. Set AlignVertically to False to align the hyperlinks in one horizontal line.

### WEViewer (ToolbarDisplay, ScrollbarsDisplay)

The WEViewer function is used to format the Crystal Reports viewer toolbar.

This function does not require a WEBUILDER function on the report.

<b>NOTE</b>	This function is a work in progress for the 4.3 DHTML viewer and is subject to change.
-------------	--

The WEViewer function syntax can be edited to create a desired look and feel for the toolbar. The export and print buttons for example can be hidden using the function syntax. The toolbar can also be set to use freeform placement using the function syntax.

There are settings in the WEViewer function syntax to alter the background colour of the viewer or to use an image as the background.

Open the WEViewer function in the **Report** menu, **Formula Workshop** for more information.

**Parameters:**

**ToolbarDisplay** accepts one of four text values. The value of the ToolbarDisplay parameter will affect the display of the Crystal Report toolbar at the top of the report viewer.

- Setting the value to “Show” will display the toolbar and display an “expand / collapse” arrow in the formula.
- Setting the value to “Hide” will hide the toolbar and display an “expand / collapse” arrow in the formula.
- Setting the value to “Suppress” will suppress the toolbar. A suppressed toolbar cannot be dynamically displayed.
- Leaving this parameter blank, i.e. “”, will display the default Crystal Report Viewer toolbar without any “Show” or “Hide” arrows.

**ScrollbarsDisplay** is not configured as of yet due to changes in the DHTML viewer.

## Inputs and Controls

This section includes functions in the InputsAndControls folder of the webElements function library. These functions are used to add common form elements to your report, such as Radio Buttons, Check boxes, and Text Areas.

### WECalculator (ElementName, ElementDefault, Validation, InvalidMessage)

The WECalculator function adds a calculator to the report. The calculator object is useful for performing quick calculations in financial reports.

#### Parameters:

**ElementName** must be set to the name of the prompt that the calculator is used to satisfy. For example, if the prompt is {?OrderAmount}, then the ElementName parameter must be set to "OrderAmount".

**ElementDefault** is a string variable representing a number. It is possible to pass a number field as a parameter if it is first converted to a string. For example `totext({InventoryAmount},0,"")`.

**Validation** is a string value that can be set to either "Numeric" or "Integer".

**InvalidMessage** is a string value that will appear as a pop-up alert if validation is used and a number has not been entered.

### WECalendar (ElementName, ElementDefault, MonthDisplays, DayDisplays)

The WECalendar function adds a calendar or a date picker control to the report.

#### Parameters:

**ElementName** must be set to the name of the datetime prompt that the calendar is used to satisfy. For example, if the prompt is {?OrderDates}, then the ElementName parameter must be set to "OrderDates".

**ElementDefault** can be any date value. The ElementDefault property indicates the starting month for the calendar.

**MonthDisplays** is the array of values that will be used on the calendar title display. If MonthDisplays is left blank, i.e. "", then the values will be "January|February|March|April|May|June|July|August|September|October|November|December".

A formula can be written to localize this string to the preferred language of the user.

**DayDisplays** is the array of values that will be used on the calendar subtitle display. If DayDisplays is left blank, i.e. "", then the default value will "Sun|Mon|Tue|Wed|Thu|Fri|Sat".

A formula can be written to localize this string to the preferred language of the user.

Ensure that the values are always 3 characters in length so that the spacing of the calendar is correct. Another option to the above is to use "Su |Mo |Tu |We |Th |Fr |Sa ". Note that in each of the strings there are three characters, two letters followed by a blank space.

### **WECalendarRange(ElementName, ElementStartDefault, ElementEndDefault, MonthDisplays, DayDisplays, AlignVertically)**

The WECalendarRange function is used to create two embedded calendars or date picker controls.

The WECalendarRange function is used in conjunction with a date range prompt.

#### **Parameters:**

**ElementName** must be set to the name of the date range prompt that the calendar is used to satisfy. For example, if the prompt is {?OrderDates}, then the ElementName parameter must be set to "OrderDates".

**ElementStartDefault** can be any date value. This will dictate the starting month for the first calendar. The ElementStartDefault parameter will be the value passed to the date prompt minimum if the user does not choose a different date.

**ElementEndDefault** can be any date value. This will dictate the starting month for the second calendar. The ElementEndDefault parameter will be the value passed to the date prompt maximum if the user does not choose a different date.

**MonthDisplays** is the array of values that will be used on the calendar title display. If MonthDisplays is left blank, i.e. "", then the values will be "January|February|March|April|May|June|July|August|September|October|November|December".

A formula can be written to localize this string to the preferred language of the user.

**DayDisplays** is the array of values that will be used on the calendar subtitle display. If DayDisplays is left blank, i.e. "", then the default value will "Sun|Mon|Tue|Wed|Thu|Fri|Sat".

A formula can be written to localize this string to the preferred language of the user.

Ensure that the values are always 3 characters in length so that the spacing of the calendar is correct. Another option to the above is to use "Su |Mo |Tu |We |Th |Fr |Sa ". Note that in each of the strings there are three characters, two letters followed by a blank space.

**AlignVertically** is a Boolean value. Setting AlignVertically to True will aligns the two calendars vertically. Setting the parameter to False will align the two calendars horizontally.

### **WECheckBox(ElementName, ElementValue, ElementDisplay, ElementDefault, AlignVertically, ElementFont, Validation, InvalidMessage)**

The WECheckBox function is used to create a check box control. This type of control allows the user to choose one or more values from a collection of checkboxes. To allow the user to select only one option, use the WERadioButton control.

The WECheckBox function can be used in conjunction with the WESelectAllClearAllReverseButtons function.

#### **Parameters:**

**ElementName** must be set to the name of the prompt that the checkbox is used to satisfy. For example, if the prompt is {?OrderOptions}, then the ElementName parameter must be set to "OrderOptions".

**ElementValue** is a character separated array of what will be passed to the URL depending on what the user selects. Separate multiple values with the "|" character.

**ElementDisplay** is a character separated array of the values that will display beside each checkbox. Separate multiple values with the "|" character.

The display is different than the value as the report developer may wish to pass an ID as a value whereas the display will show the name associated with that particular ID. If no display is required beside the check box use "".

**ElementDefault** can be one or more values or left blank. All default values will be checked when the report is first viewed. If the prompt is used as the default, e.g. {?Country}, the value or values selected will be checked when the target report is opened.

**AlignVertically** is a Boolean value. Setting AlignVertically to True aligns the checkboxes and display values vertically. Setting this value to False places the checkboxes and display values in one horizontal line.

**ElementFont** is used to format the text of the display elements. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style. **Note:** This is a single value that will format all display elements generated by this function.

**Validation** should be set to "Checked" if validation is required for this function.

**InvalidMessage** is the message that will be displayed in an alert pop-up if no boxes are checked at the time of submission. Validation must be set to "Checked" in order for the message to display.

### **WEComboBoxSelect (ElementName, ElementValues, ElementDisplays, ElementDefault, AllowCustomValues, MatchSelectValues, ElementBoxSize, ElementSelectSize, AlignVertically, ElementFont, Validation, InvalidMessage)**

The WEComboBoxSelect function is used to create a combination select text box control. This type of control allows the user to choose a single value from a set of choices or to enter the value in a text box. Values entered into the text box will update the default value in the select menu.

**Parameters:**

**ElementName** must be set to the name of the prompt that the drop down menu is used to satisfy. For example, if the prompt is {?OrderOptions}, then the ElementName parameter must be set to "OrderOptions".

**ElementValues** is a character separated array of what will be passed to the URL depending on what the user selects. Separate multiple values with the "|" character.

**ElementDisplays** is a character separated array of the values that will display in the drop down list. Separate multiple values with the "|" character.

The display is different than the value as the report developer may wish to pass an ID as a value whereas the display will show the name associated with that particular ID.

**ElementDefault** is a single text value that will appear at the top of the drop-down menu and in the text box.

**AllowCustomValues** is a Boolean value. Setting AllowCustomValues to True gives the end user the ability to enter a value in the text input box that is not a value in the select menu. Setting this value to False forces the end user to enter a value into the text input box that matches a value in the select menu. This is done by a character by character input match.

**MatchSelectValues** is a Boolean value. Setting MatchSelectValues to True will match the text box input to a value in the select menu versus matching the select menu display.

For example, if MatchSelectValues was True the end user could type a Customer ID into the text input box. The select menu would change to display the Customer Name which corresponds to that ID.

**ElementBoxSize** is the length, in characters, of the text box input.

**ElementSelectSize** is the height, in number of lines, of the select menu.

**AlignVertically** is a Boolean value. Setting AlignVertically to True aligns the text box input and select menu vertically. Setting this value to False places the text box input and select menu in one horizontal line.

**ElementFont** is used to format the text of the drop down list and the text of the text box. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style. **Note:** This is a single value that will format all display elements generated by this function.

**Validation** is used to ensure that the user selects a value from the dropdown menu or types in a valid value in the text box before the URL is run. Please see the Validation section in the webElements Parameter List for more information on validation.

**InvalidMessage** is the message that will be displayed in an alert pop-up if no value is selected at the time of submission.

### **WERadio (ElementName, ElementValues, ElementDisplays, ElementDefault, AlignVertically, ElementFont, Validation, InvalidMessage)**

The WERadio function is used to create a radio button control. This type of control allows the user to choose only one value from a set of choices. To allow the user to select more than once option, use the WECheckBox control.

**Parameters:**

**ElementName** must be set to the name of the prompt that the radio button is used to satisfy. For example, if the prompt is {?OrderOptions}, then the ElementName parameter must be set to "OrderOptions".

**ElementValues** is a character separated array of what will be passed to the URL depending on what the user selects. Separate multiple values with the "|" character.

**ElementDisplays** is a character separated array of the values that will display beside each checkbox. Separate multiple values with the "|" character.

The display is different than the value as the report developer may wish to pass an ID as a value whereas the display will show the name associated with that particular ID. If no display is required beside the check box use "".

**ElementDefault** is a single text value that is selected by default.

If the prompt is used as the default, e.g. {?Country}, the value selected will be checked when the target report is opened.

**AlignVertically** is a Boolean value. Setting AlignVertically to True aligns the radio buttons and display values vertically. Setting this value to False or leaving the parameter empty places the radio buttons and display values in one horizontal line.

**ElementFont** is used to format the text of the display elements. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style. **Note:** This is a single value that will format all URLs generated by this function.

**Validation** should be set to "Checked" if validation is required for this function.

**InvalidMessage** is the message that will be displayed in an alert pop-up if no radio buttons are selected at the time of submission. Validation must be set to "Checked" in order for the message to display.

**WESelect (ElementName, ElementValues, ElementDisplays, ElementDefault, ElementFont)**

The WESelect function is used to create a drop-down menu. This type of control allows the user to choose a single value from a set of choices.

**Parameters:**

**ElementName** must be set to the name of the prompt that the drop down menu is used to satisfy. For example, if the prompt is {?OrderOptions}, then the ElementName parameter must be set to "OrderOptions".

**ElementValues** is a character separated array of what will be passed to the URL depending on what the user selects. Separate multiple values with the "|" character.

**ElementDisplays** is a character separated array of the values that will display in the drop down list. Separate multiple values with the "|" character.

The display is different than the value as the report developer may wish to pass an ID as a value whereas the display will show the name associated with that particular ID.

**ElementDefault** is a single text value that will appear at the top of the drop-down menu.



**ElementFont** is used to format the text of the drop down list. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style. **Note:** This is a single value that will format all display elements generated by this function.

<b>NOTE</b>	There is no validation for Weselect. If nothing is selected by the user, the first value on the list will be passed to the URL.
-------------	---

### **WeselectCascade (ElementName, ElementDisplays, ElementSet, ElementDefault, ElementConstant, Prompt, ElementWidth, ElementFont, Validation, InvalidMessage)**

The WeselectCascade function is used to create a drop-down menu that is part of a cascading prompt set. This type of control allows the user to choose a single value from a set of choices. To allow the user to select more than one value, use the WeselectMultiCascade function.

#### **Parameters:**

**ElementName** must be set to the name of the prompt that the cascading list is used to satisfy. For example, if the prompt is {?OrderOptions}, then the ElementName parameter must be set to "OrderOptions".

**ElementDisplays** is an array of what the user will see in the drop-down menu. The chosen display will also be the value passed via the URL.

A first level displays array would have syntax such as "Level 1 Value 1|Level 1 Value 2|Level 1 Value 3"

The following is an example of a displays array for a first level prompt in a Region|Country|City set.

```
"North America|South America|Europe"
```

A second or greater level displays array would have syntax such as

```
"|Level 1 Value 1|Level 2 Value 1|Level 2 Value 2||Level 1 Value 2|Level 2 Value 3"
```

The following is an example of a displays array for a second level prompt in a Region|Country|City set.

```
"|North America|Canada|USA|Mexico||South America|Brazil|Argentina||Europe|France|Germany|United Kingdom"
```

An example of a displays array for a third level prompt in a Region|Country|City set would be

```
"|BC|Vancouver||CA|San Francisco|San Jose||FLA|Miami"
```

**ElementSet** is the character separated prompt names in a set or family of cascading prompts. For example, if a set of prompts "Region", "Country", "City" is on the report, the ElementSet name for all of the controls would be "Region|Country|City".

The ElementSet name must be identical for all prompts that are in the same cascade set or family or they will not cascade properly.

There can be more than one set or family of cascading prompts on a report. For example there can be an ElementSet for a product hierarchy

“ProductCategory|ProductSubcategory|ProductName” and an ElementSet for a geographical hierarchy “Region|Country|City” in the same report. These separate sets of prompts will not influence each other.

**ElementDefault** is a single text value that will appear at the top of the drop-down menu.

If the prompt is used as the default, e.g. {?Country}, the value selected will appear at the top of the menu when the target report is opened.

**ElementConstant** is an option value that will appear at the top of the drop-down menu at all times. This is useful for keeping a wildcard value such as “All” available for the report user.

If the report user does not choose an option in the menu, then the ElementConstant will be passed automatically to the URL.

**Prompt** is a Boolean value. Setting Prompt to True will append the chosen value along with the ElementName to the URL. This requires that a prompt of the same name be present on the report. Setting Prompt to False will allow the control to cascade but the ElementName and chosen value will not be appended to the URL.

**ElementWidth** represents the width of the control in characters. Ensure that the width set in the formula will be large enough to display the longest option.

**ElementFont** is used to format the text in the drop-down menu. If ElementFont is left blank, i.e. “”, the font will assume the formatting of the formula object containing the WESelectCascade function.

**Validation** is used to ensure that the user selects a value from the dropdown menu before the URL is run. Please see the Validation section in the webElements Parameter List for more information on validation.

**InvalidMessage** is the message that will be displayed in an alert pop-up if no values are selected at the time of submission.

<b>NOTE</b>	If no ElementConstant is used then there is a risk of having no prompt value passed to the URL unless the user enters values for every select menu in the prompt family.
-------------	--

### **WESelectCascadeExt (ElementName, ElementValues, ElementDisplays, ElementSet, ElementDefault, ElementConstant, Prompt, ElementWidth, ElementFont, Validation, InvalidMessage)**

The WESelectCascadeExt function is used to create a drop-down menu that is part of a cascading prompt set. This type of control allows the user to choose a single value from a set of choices.

The WESelectCascadeExt differs from the WESelectCascade function in that it allows for a different value to be passed to the URL than the display value. For example, a customer names could appear in the drop down menu while a customer ID is passed to the URL.

#### **Parameters:**

**ElementName** must be set to the name of the prompt that the cascading list is used to satisfy. For example, if the prompt is {?OrderOptions}, then the ElementName parameter must be set to “OrderOptions”.

**ElementValues** is an array of values that may be passed to the URL.

A first level values array would have syntax such as

```
"Level 1 ID 1|Level 1 ID 2|Level 1 ID 3"
```

The following is an example of a displays array for a first level prompt in a RegionID|CountryID|CityID set.

```
"101|102|201"
```

A second or greater level displays array would have syntax such as

```
"|Level 1 Value 1|Level 2 Value 1|Level 2 Value 2||Level 1 Value 2|Level 2 Value 3"
```

The following is an example of a values array for a second level prompt in a Region|Country|City set.

```
"|101|1011|1012|1013||102|1021|1022||201|2011|2012|2013"
```

An example of a values array for a third level prompt in a Region|Country|City set would be

```
"|1011|10111||1012|10121|10122|10123..."
```

**ElementDisplays** is an array of what the user will see in the drop-down menu. The chosen display will also be the value passed via the URL.

A first level displays array would have syntax such as "Level 1 Value 1|Level 1 Value 2|Level 1 Value 3"

The following is an example of a displays array for a first level prompt in a Region|Country|City set.

```
"North America|South America|Europe"
```

A second or greater level displays array would have syntax such as

```
"|Level 1 Value 1|Level 2 Value 1|Level 2 Value 2||Level 1 Value 2|Level 2 Value 3"
```

The following is an example of a displays array for a second level prompt in a Region|Country|City set.

```
"|North America|Canada|USA|Mexico||South America|Brazil  
Argentina||Europe|France|Germany|United Kingdom"
```

An example of a displays array for a third level prompt in a Region|Country|City set would be

```
"|BC|Vancouver||CA|San Francisco|San Jose||FLA|Miami"
```

**ElementSet** is the character separated prompt names in a set or family of cascading prompts. For example, if a set of prompts "Region", "Country", "City" is on the report, the ElementSet name for all of the controls would be "Region|Country|City".

The ElementSet name must be identical for all prompts that are in the same cascade set or family or they will not cascade properly.

There can be more than one set or family of cascading prompts on a report. For example there can be an ElementSet for a product hierarchy

“ProductCategory|ProductSubcategory|ProductName” and an ElementSet for a geographical hierarchy “Region|Country|City” in the same report. These separate sets of prompts will not influence each other.

**ElementDefault** is a single text value that will appear at the top of the drop-down menu.

If the prompt is used as the default, e.g. {?Country}, the value selected will appear at the top of the menu when the target report is opened.

**ElementConstant** is an option value that will appear at the top of the drop-down menu at all times. This is useful for keeping a wildcard value such as “All” available for the report user.

If the report user does not choose an option in the menu, then the ElementConstant will be passed automatically to the URL.

**Prompt** is a Boolean value. Setting Prompt to True will append the chosen value along with the ElementName to the URL. This requires that a prompt of the same name be present on the report. Setting Prompt to False will allow the control to cascade but the ElementName and chosen value will not be appended to the URL.

**ElementWidth** represents the width of the control in characters. Ensure that the width set in the formula will be large enough to display the longest option.

**ElementFont** is used to format the text in the drop-down menu. If ElementFont is left blank, i.e. “”, the font will assume the formatting of the formula object containing the WESelectCascade function.

**Validation** is used to ensure that the user selects a value from the dropdown menu before the URL is run. Please see the Validation section in the webElements Parameter List for more information on validation.

**InvalidMessage** is the message that will be displayed in an alert pop-up if no values are selected at the time of submission.

<b>NOTE</b>	If no ElementConstant is used then there is a risk of having no prompt value passed to the URL unless the user enters values for every select menu in the prompt family.
-------------	--

### **WESelectDate(ElementName, ElementDefault, MonthDisplays, StartYear, EndYear, ElementFont)**

The WESelectDate function is used to create a set of three drop-down menus to select a year, month, and day.

#### **Parameters:**

**ElementName** must be set to the name of the prompt that the drop-down menu is used to satisfy. For example, if the prompt is {?OrderOptions}, then the ElementName parameter must be set to “OrderOptions”.

**ElementDefault** is a date value that will appear at the top of each drop-down menu when the report is viewed.

If the prompt is used as the default, such as {?StartDate}, the date selected will appear at the top of the menus when the target report is viewed.

**MonthDisplays** is a character separated array of what the user will see in the drop-down menu for the months. This character separated array must include twelve (12) values.

If nothing is entered as a month display, i.e. "", then the function will use the default value of:

"January|February|March|April|May|June|July|August|September|October|November|December"

**StartYear** is a number representing the minimum available year.

**EndYear** is a number representing the maximum available year.

**ElementFont** is used to format the text in the drop-down menu. If ElementFont is left blank, i.e. "", the font will assume the formatting of the formula object containing the WESelectDate function.

<b>NOTE</b>	There is no validation for WESelectDate. If nothing is selected by the user, the first value on the list will be passed to the URL.
-------------	---

### **WESelectDateRange(ElementName, ElementStartDefault, ElementEndDefault, MonthDisplays, StartYear, EndYear, ElementFont, AlignVertically)**

The WESelectDate function is used to create a set of three drop-down menus to select a year, month, and day.

WESelectDateRange is used in conjunction with a date range prompt.

#### **Parameters:**

**ElementName** must be set to the name of the prompt that the drop-down menu is used to satisfy. For example, if the prompt is {?OrderOptions}, then the ElementName parameter must be set to "OrderOptions".

**ElementStartDefault** is a date value that will appear at the top of the first set of drop-down menus when the report is viewed.

If the prompt is used as the default, such as {?StartDate}, the date selected will appear at the top of the menus when the target report is viewed.

**ElementEndDefault** is a date value that will appear at the top of the second set of drop-down menus when the report is viewed.

If the prompt is used as the default, e.g. {?EndDate}, the date selected will appear at the top of the second set of menus when the target report is viewed.

**MonthDisplays** is a character separated array of what the user will see in the drop-down menu for the months. This character separated array must include twelve (12) values.

If nothing is entered as a month display, i.e. "", then the function will use the default value of:

"January|February|March|April|May|June|July|August|September|October|November|December"

**StartYear** is a number representing the minimum available year.

**EndYear** is a number representing the maximum available year.

**ElementFont** is used to format the text in the drop-down menu. If ElementFont is left blank, i.e. "", the font will assume the formatting of the formula object containing the WeselectDateRange function.

**AlignVertically** is a Boolean value. Setting this parameter to True aligns the two sets of date select prompts vertically. Setting this parameter to False places the controls in one horizontal line.

**NOTE**

There is no validation for WeselectDateRange. If nothing is selected by the user, the first value on the list will be passed to the URL.

### WeselectDuo (ElementName, AValues, ADisplays, BValues, BDisplays, ElementDefault, ElementFont)

The WeselectDuo function is used to create two drop-down menus. When the page is submitted, the values from both drop-down menus will be concatenated and passed as a single parameter to the URL.

For example WeselectDuo could be used to create Year and Quarter controls where the final result passed to the URL will be YYQ.

**Parameters:**

**ElementName** must be set to the name of the prompt that the drop-down menus satisfy. For example, if the prompt is {?OrderOptions}, then the ElementName parameter must be set to "OrderOptions".

**AValues** is a character separated array of values for the first drop-down menu. An example of a values array would be "02|03|04|05|06". Separate multiple values with the "|" character.

**ADisplays** is a character separated array of the values that will display in the first checkbox. Separate multiple values with the "|" character.

The display is different than the value as the report developer may wish to pass an ID as a value whereas the display will show the name associated with that particular ID.

**BValues** is a character separated array of values for the second drop-down menu. An example of a values array would be "1 | 2 | 3 | 4". Separate multiple values with the "|" character.

**BDisplays** is a character separated array of the values that will display in the second checkbox. Separate multiple values with the "|" character.

The display is different than the value as the report developer may wish to pass an ID as a value whereas the display will show the name associated with that particular ID.

**ElementDefault** is a character separated pair of values that will appear at the top of the first and second controls.

If you want to display the chosen prompt values as the default, e.g. {? YearQuarter}, you would use syntax similar to

```
{?YearQuarter}[1 to 2] + "|" + {?YearQuarter}[3]
```

**ElementFont** is used to format the text of the drop-down list elements. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style. **Note:** This is a single value that will format all display elements generated by this function.

<b>NOTE</b>	There is no validation for WeselectDuo. If nothing is selected by the user, the first values on the controls will be passed to the URL.
-------------	---

### WeselectMulti (ElementName, ElementValues, ElementDisplays, ElementDefault, ElementSize, ElementFont)

The WeselectMulti function is used to create a list of items. This type of control allows the user to choose one or more values from a set of choices.

#### Parameters:

**ElementName** must be set to the name of the prompt that the multi-select control is used to satisfy. For example, if the prompt is {?OrderOptions}, then the ElementName parameter must be set to "OrderOptions".

**ElementValues** is a character separated array of what will be passed to the URL depending on what the user selects. Separate multiple values with the "|" character.

**ElementDisplays** is a character separated array of the values that will display in the drop down list. Separate multiple values with the "|" character.

The display is different than the value as the report developer may wish to pass an ID as a value whereas the display will show the name associated with that particular ID.

**ElementDefault** is a single text value that will appear selected by default.

**ElementSize** is the height, in number of lines, of the multi-select control.

**ElementFont** is used to format the text in the multi-select control. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style. **Note:** This is a single value that will format all display elements generated by this function.

<b>NOTE</b>	There is no validation for WeselectMulti. If nothing is selected by the user, the first value on the list will be passed to the URL.
-------------	--

### WeselectMultiCascade(ElementName, ElementDisplays, ElementSet, ElementDefault, ElementConstant, Prompt, ElementWidth, ElementSize, ElementFont, Validation, InvalidMessage)

The WeselectMultiCascade function creates a drop-down menu that is part of a cascading prompt set. This type of control allows the user to choose more than one value from a set of choices. To allow the user to select only one value, use the WeselectCascade function.

#### Parameters:

**ElementName** must be set to the name of the prompt that the cascading list is used to satisfy. For example, if the prompt is {?OrderOptions}, then the ElementName parameter must be set to "OrderOptions".

**ElementDisplays** is an array of what the user will see in the drop-down menu. The chosen display will also be the value passed via the URL.

A first level displays array would have syntax such as “Level 1 Value 1|Level 1 Value 2|Level 1 Value 3”

The following is an example of a displays array for a first level prompt in a Region|Country|City set.

```
“North America|South America|Europe”
```

A second or greater level displays array would have syntax such as

```
“| |Level 1 Value 1|Level 2 Value 1|Level 2 Value 2| |Level 1 Value 2|Level 2 Value 3”
```

The following is an example of a displays array for a second level prompt in a Region|Country|City set.

```
“| |North America|Canada|USA|Mexico| |South America|Brazil  
Argentina| |Europe|France|Germany|United Kingdom”
```

An example of a displays array for a third level prompt in a Region|Country|City set would be

```
“| |BC|Vancouver| |CA|San Francisco|San Jose| |FLA|Miami”
```

**ElementSet** is the character separated prompt names in a set or family of cascading prompts. For example, if a set of prompts “Region”, “Country”, “City” is on the report, the ElementSet name for all of the controls would be “Region|Country|City”.

The ElementSet name must be identical for all prompts that are in the same cascade set or family or they will not cascade properly.

There can be more than one set or family of cascading prompts on a report. For example there can be an ElementSet for a product hierarchy “ProductCategory|ProductSubcategory|ProductName” and an ElementSet for a geographical hierarchy “Region|Country|City” in the same report. These separate sets of prompts will not influence each other.

**ElementDefault** is a single text value that will appear at the top of the drop-down menu.

If the prompt is used as the default, e.g. {?Country}, the value selected will appear at the top of the menu when the target report is opened.

**ElementConstant** is an option value that will appear at the top of the drop-down menu at all times. This is useful for keeping a wildcard value such as “All” available for the report user.

If the report user does not choose an option in the menu, then the ElementConstant will be passed automatically to the URL.

**Prompt** is a Boolean value. Setting Prompt to True will append the chosen value along with the ElementName to the URL. This requires that a prompt of the same name be present on the report. Setting Prompt to False will allow the control to cascade but the ElementName and chosen value will not be appended to the URL.

**ElementWidth** represents the width of the control in characters. Ensure that the width set in the formula will be large enough to display the longest option.



**ElementSize** is the height, in number of lines, of the control.

**ElementFont** is used to format the text in the drop-down menu. If ElementFont is left blank, i.e. "", the font will assume the formatting of the formula object containing the WSelectMultiCascade function.

**Validation** is used to ensure that the user selects a value from the dropdown menu before the URL is run. Please see the Validation section in the webElements Parameter List for more information on validation.

**InvalidMessage** is the message that will be displayed in an alert pop-up if no values are selected at the time of submission.

<b>NOTE</b>	There is validation available for WSelectMultiCascade function. If no ElementConstant is defined then there is a risk of having no prompt value passed to the URL unless the user enters values for all select menus in the prompt family.
-------------	--

### **WSelectMultiCascadeExt(ElementName, ElementValues, ElementDisplays, ElementSet, ElementDefault, ElementConstantValue, ElementConstantDisplay, Prompt, ElementWidth, ElementSize, ElementFont, Validation, InvalidMessage)**

The WSelectMultiCascadeExt function is used to create a drop-down menu that is part of a cascading prompt set. This type of control allows the user to choose a single value from a set of choices.

The WSelectMultiCascadeExt function differs from the WSelectMultiCascade function in that it allows for a different value to be passed to the URL than the display value. For example, a customer name appears in the drop down menu while a customer ID is passed to the URL.

#### **Parameters:**

**ElementValues** is an array of values that may be passed to the URL.

A first level values array would have syntax such as

```
"Level 1 ID 1|Level 1 ID 2|Level 1 ID 3"
```

The following is an example of a displays array for a first level prompt in a RegionID|CountryID|CityID set.

```
"101|102|201"
```

A second or greater level displays array would have syntax such as

```
"||Level 1 Value 1|Level 2 Value 1|Level 2 Value 2||Level 1 Value 2|Level 2 Value 3"
```

The following is an example of a values array for a second level prompt in a Region|Country|City set.

```
"||101|1011|1012|1013||102|1021|1022||201|2011|2012|2013"
```

An example of a values array for a third level prompt in a Region|Country|City set would be

```
"|1011|10111|1012|10121|10122|10123..."
```

**ElementName** must be set to the name of the prompt that the cascading list is used to satisfy. For example, if the prompt is {?OrderOptions}, then the ElementName parameter must be set to "OrderOptions".

**ElementDisplays** is an array of what the user will see in the select multi menu. The chosen display will also be the value passed via the URL.

A first level ElementDisplays array would have syntax such as

```
"Level 1 Value 1|Level 1 Value 2|Level 1 Value 3"
```

The following is an example of a displays array for a first level prompt in a Region|Country|City set.

```
"North America|South America|Europe"
```

A second or greater level displays array would have syntax such as

```
"|Level 1 Value 1|Level 2 Value 1|Level 2 Value 2||Level 1 Value 2|Level 2 Value 3"
```

The following is an example of a displays array for a second level prompt in a Region|Country|City set.

```
"|North America|Canada|USA|Mexico||South America|Brazil  
Argentina||Europe|France|Germany|United Kingdom"
```

An example of a displays array for a third level prompt in a Region|Country|City set would be

```
"|Canada|Vancouver||USA|San Francisco|San Jose|Miami..."
```

**ElementSet** is the character separated "|" prompt names in a set or family of cascading prompts. For example, if a set of prompts "Region", "Country", "City" is on the report, the ElementSet name for all of the controls would be "Region|Country|City".

The ElementSet name must be identical for all prompts that are in the same cascade set or family or they will not cascade properly.

There can be more than one set or family of cascading prompts on a report. For example there can be an ElementSet for a product hierarchy "ProductCategory|ProductSubcategory|ProductName" and an ElementSet for a geographical hierarchy "Region|Country|City" on the same report. These separate prompt sets or families will not influence each other.

**ElementDefault** is a single text value that will appear at the top of the select multi menu.

If the prompt is used as the default, e.g. {?Country}, the value selected will appear at the top of the menu when the target report is opened.

**ElementConstantValue** is the value that corresponds to the ElementConstantDisplay.

If the report user does not choose an option in the menu, then the ElementConstantValue will be passed automatically to the URL.

**ElementConstantDisplay** is an option value that will appear at the top of the drop-down menu at all times. This is useful for keeping a wildcard value such as “All” available for the report user.

**Prompt** is a Boolean value. Setting Prompt to True will append the chosen value along with the ElementName to the URL. This requires that a prompt of the same name be present on the report. Setting Prompt to False will allow the control to cascade but the ElementName and chosen value will not be appended to the URL.

**ElementWidth** represents the width of the control in characters as set by the report developer at design time. Ensure that the width set in the formula will be great enough to display the longest option.

**ElementSize** is the height, in number of lines, which will be displayed on the report.

**ElementFont** is used to format the text in the drop-down menu...this can be left blank (""), and the font will assume the font formatting of the formula object containing the WESelectCascade element, or the report developer can use the WEFonts function in the Formatting section to easily define a formatted font style.

**Validation** is used to ensure that the user selects a value from the dropdown menu before the URL is run. Please see the Validation section in the webElements Parameter List for more information on validation.

**InvalidMessage** is the message that will be displayed in an alert pop-up if no values are selected at the time of submission.

<b>NOTE</b>	There is validation available for WESelectMultiCascadeExt. If no ElementConstantValue is used then there is a risk of having no prompt value passed to the URL unless the user enters values for all select menus in the prompt family.
-------------	---

### **WESelectTrio (ElementName, AValues, ADisplays, BValues, BDisplays, CValues, CDisplays, ElementDefault, ElementFont)**

The WESelectTrio function is used to create three drop-down menus. When the page is submitted, the values from all three drop-down menus will be concatenated and passed as a single parameter to the URL.

For example WESelectTrio could be used to create Location, Division, Group controls where the final result passed to the URL will be a single ID field representing location, division, and group.

#### **Parameters:**

**ElementName** must be set to the name of the prompt that the drop-down menus satisfy. For example, if the prompt is {?OrderOptions}, then the ElementName parameter must be set to “OrderOptions”.

**AValues** is a character separated array of values for the first drop-down menu. An example of a values array would be “100|101|102”. Separate multiple values with the “|” character.

**ADisplays** is a character separated array of the values that will display in the first checkbox. Separate multiple values with the “|” character.

The display is different than the value as the report developer may wish to pass an ID as a value whereas the display will show the name associated with that particular ID.

**BValues** is a character separated array of values for the second drop-down menu. Separate multiple values with the “|” character.

**BDisplays** is a character separated array of the values that will display in the second checkbox. Separate multiple values with the “|” character.

**CValues** is a character separated array of values for the second drop-down menu. Separate multiple values with the “|” character.

**CDisplays** is a character separated array of the values that will display in the second checkbox. Separate multiple values with the “|” character.

The display is different than the value as the report developer may wish to pass an ID as a value whereas the display will show the name associated with that particular ID.

**ElementDefault** is a character separated trio of values that will appear at the top of the individual select menus.

If you want to display the chosen prompt values as the default, e.g. {? DepartmentCode}, you would use syntax similar to

```
{?DepartmentCode}[1 to 3] + "|" + {?DepartmentCode}[4] + "|" +
{?DepartmentCode}[5]
```

**ElementFont** is used to format the text of the drop-down list elements. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style. **Note:** This is a single value that will format all display elements generated by this function.

**Example:**  
An example of a WESelectTrio function with parameter values entered would be

```
stringvar avalues:= "101|102|103";
stringvar adisplays:= "Location 1|Location 2|Location 3";
stringvar bvalues:= "a|b|c|d";
stringvar bdisplays:= "Group A|Group B|Group C|Group D";
stringvar cvalues:= "s|m|f";
stringvar cdisplays:= "Sales Department|Marketing Department|Finance Department";
stringvar defaultvalue:= {?DepartmentCode}[1 to 3] + "|" + {?DepartmentCode}[4] + "|" +
{?DepartmentCode}[5];
WESelectTrio ("DepartmentCode", avalues, adisplays, bvalues, bdisplays, cvalues, cdisplays,
defaultvalue, "") +
```

NOTE	There is no validation for WESelectTrio. If nothing is selected by the user, the first values on the controls will be passed to the URL.
------	--

### WESpin (ElementName, ElementDefault, ElementMin, ElementMax, IncrementValue, ElementSize, ElementFont, Validation, ValidationMessage)

The WESpin function is used to create a “spinner” control. This type of control allows the user to increment and select a number.

#### Parameters:

**ElementName** must be set to the name of the prompt that the drop down menu is used to satisfy. For example, if the prompt is {?SalesLimit}, then the ElementName parameter must be set to “SalesLimit”.

**ElementDefault** is a numeric value that will appear in the text box of the control.

**ElementMin** is the lower limit of the control. The end user cannot use the decrement button to produce a lower value than this in the text box.

**ElementMax** is the upper limit of the control. The end user cannot use the increment button to produce a higher value than this in the text box.

**IncrementValue** is used to set how the number in the text box increments or decrements. For example, if IncrementValue was set to 5, every time the end user touched the increment button, the value in the text box would increase by 5.

**ElementSize** is the length, in characters, of the text box in the control.

**ElementFont** is used to format the text in the text box and the buttons. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style.

**Validation** is used to ensure that the user enters a value into the text box before the URL is run. This should be set to “Number” if validation is required.

**InvalidMessage** is the message that will be displayed in an alert pop-up if there is not a valid value in the control.

### WETextArea (ElementName, ElementDefault, ElementHeight, ElementWidth, ElementFont, Validation, InvalidMessage)

The WETextArea function is used to create a text area input control. A text area is a multi line text input field control that allows text to wrap and scroll. For a single line text input, use the WETextBox function.

#### Parameters:

**ElementName** must be set to the name of the prompt that the text area satisfies. For example, if the prompt is {?OrderOptions}, then the ElementName parameter must be set to “OrderOptions”.

**ElementDefault** is the text that you wish to display in the text area when the report is opened.

If the prompt is used as the default, e.g. {?Country}, the value entered will appear in the text box when the target report is opened.

**ElementHeight** is a text value that can be in inches (e.g. "3in") centimeters (e.g. "6cm"), or pixels (e.g. "1600pi").

**ElementWidth** is a text value that can be in inches (e.g. "3in") centimeters (e.g. "6cm"), or pixels (e.g. "1600pi").

**ElementFont** is used to format the text in the text area. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style.

**Validation** is used to ensure that the user enters a value into the text area before the URL is run.

**InvalidMessage** is the message that will be displayed in an alert pop-up if the user does not enter valid values into the text area.

### **WETextBox(ElementName, ElementDefault, ElementSize, MaxLength, ElementFont, Validation, InvalidMessage)**

The WETextBox function is used to create a text box control within the report. A text box control allows the user to enter custom text. Text box controls are one line high – for a text field that is more than a one line high, use the WETextArea function.

#### **Parameters:**

**ElementName** must be set to the name of the prompt that the text box satisfies. For example, if the prompt is {?OrderOptions}, then the ElementName parameter must be set to "OrderOptions".

**ElementDefault** is the text that you wish to display in the text box when the report is opened.

If the prompt is used as the default, e.g. {?Country}, the value entered will appear in the text box when the target report is opened.

**MaxLength** is the maximum number of characters that a user is allowed to type into the text box control.

**ElementSize** is the length, in characters, of the text box control.

**ElementFont** is used to format the text in the text box. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style.

**Validation** is used to ensure that the user enters a value into the text box before the URL is run.

**InvalidMessage** is the message that will be displayed in an alert pop-up if the user does not enter valid values into the textbox.

#### **Example:**

An example of a WETextBox function with parameter values entered would be

```
WETextBox("Country", "Canada", 16, 16, {?Country}, "Empty", "Please enter the Country here.")
```

### **WETextBoxAndCheckBox(ElementNameTextbox, ElementDisplayTextbox, ElementSizeTextbox, MaxLengthTextbox, ElementFontTextbox, ElementNameCheckbox, ElementValueCheckbox, ElementDefaultCheckbox)**

The WETextBoxAndCheckBox function is used to create a text input that is bound to a check box control. This type of control allows the user to choose one or more values from a set of choices. The value of both the checkbox and the textbox are passed to the URL.

The WETextBoxAndCheckBox function can be used in conjunction with the WESelectAllClearAllReverseButtons function.

#### **Parameters:**

**ElementNameTextbox** must be the name of the prompt that the textbox is used to satisfy. i.e. if the prompt is {?Country}, it is imperative that the ElementNameTextbox be "Country".

**ElementDisplayTextbox** is the text that will appear beside the text box control.

**ElementSizeTextbox** is the length, in characters, of the textbox input.

**MaxLengthTextbox** is the maximum acceptable length of the input string.

**ElementFontTextbox** is used to format the text that is typed and displayed in the text box. If this is the font will assume the formatting of the formula object containing the WETextBoxAndCheckBox element.

**ElementNameCheckbox** must be the name of the prompt that the checkbox is used to satisfy. For example, if the prompt is {?Country}, it is imperative that the ElementNameCheckbox be "Country".

**ElementValueCheckbox** are the values that will be passed to the URL depending on which checkboxes are selected by the user.

**ElementDefaultTextbox** is a String value that will appear in the textbox by default. This value can be blank, a static value, a field or a formula.

### **WETextBoxMulti (ElementName, ElementDefault, ElementWidth, MaxLength, TextAreaRows, ElementFont, ElementLabels, Validation, InvalidMessage)**

The WETextBoxMulti function is used to create a multiple value text box control within the report. A text box control allows the user to enter multiple values of custom text.

#### **Parameters:**

**ElementName** must be set to the name of the prompt that the text box satisfies. For example, if the prompt is {?OrderOptions}, then the ElementName parameter must be set to "OrderOptions".

**ElementDefault** is the text values that you wish to display in the multiple value text box when the report is opened.

If a prompt is used as the default, with syntax similar to join({?Countries},"|"), the values entered will appear in the text box when the target report is opened.

**ElementWidth** is a text value that can be in inches (e.g. "3in") centimeters (e.g. "6cm"), or pixels (e.g. "1600pi").

**MaxLength** is the maximum number of characters that a user is allowed to type into the text box control.

**TextAreaRows** is the height, in number of rows, of the multiple value text box control. If the number of values entered into the control is greater than TextAreaRows, then the control will have a vertical scrollbar.

Note that TextAreaRows does not limit the number of values that can be added to the control.

**ElementFont** is used to format the text in the text box. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style.

**ElementLabels** is a character separated value used to define the "Add", "Reset" and "Clear" label links at the bottom of the text area. For example "Add Value|Reset Control|Clear All".

If this parameter is left blank then the defaults of Add, Reset and Clear are used.

**Validation** is used to ensure that the user enters a value into the text box before the URL is run.

**InvalidMessage** is the message that will be displayed in an alert pop-up if the user does not enter valid values into the textbox.

#### Example:

An example of a WETextBoxMulti function with parameter values entered would be

```
stringvar elementdefault:= join({?Countries},"|");
stringvar elementfont:= WEFont ("Verdana", 8, "green", "left", false, false, "", "");
WETextBoxMulti ("Countries", elementdefault, 16, 16, 3, elementfont, "", "empty", "enter a
value please")
```

### WETreePicker(ElementName, ElementValues, ElementHierarchy, ElementPicks, ElementDefault, TextAreaHeight, TextAreaWidth, TextAreaFont, ElementLabels, Validation, InvalidMessage)

The WETreePicker function is used to create a tree picker control in a report. The tree menu can have multiple levels of hierarchy. Values picked from the tree are inserted into a text area.

The WETreePicker function syntax can be edited to create a desired look and feel for your tree menu. Menu fonts and icons can be customized within the function syntax. Open the WETreePicker in **Report > Formula Workshop** for more information.

#### Parameters:

**ElementName** must be set to the name of the prompt that the tree picker control satisfies. For example, if the prompt is {?groups}, then the ElementName parameter must be set to "groups".



**ElementValues** is a character separated array of what will be displayed in the tree menu. Values can include text objects (using quoted text or WETextObject functions) for folders and values.

Separate multiple values with the "|" character.

**ElementHierarchy** is a character separated array representing the hierarchy of the ElementValues of the tree menu. A zero (0) represents a parent or root folder level item where a one (1) represents the second level of hierarchy and a two (2) represents the third level of hierarchy.

Separate multiple values with the "|" character.

**ElementPicks** is a character separated array representing the values of the tree menu that are available to be chosen by the end user. A yes (y) represents a value that can be added into the chosen value text area. Element picks can be either a folder or child item. A no (n) represents a folder or value in the tree that cannot be added into the chosen value text area.

Separate multiple values with the "|" character.

**ElementDefault** is the value or values that you wish to display in the text area when the report is opened.

Separate multiple values with the "|" character.

If the prompt is used as the default, e.g. split({?Country},"|"), the values entered will appear in the text box when the target report is opened.

**TextAreaHeight** is a text value that can be in inches (e.g. "3in") centimeters (e.g. "6cm"), or pixels (e.g. "1600pi").

**TextAreaWidth** is a text value that can be in inches (e.g. "3in") centimeters (e.g. "6cm"), or pixels (e.g. "1600pi").

**TextAreaFont** is used to format the text in the text area. The report developer can use the WEFont function in the Formatting functions to easily define a formatted font style.

**ElementLabels** is a character separated value used to define the "Reset" and "Clear" label links at the bottom of the text area. For example "Reset Choices|Clear Choices".

If this parameter is left blank then the defaults of Reset and Clear are used.

**Validation** is used to ensure that the user enters a value into the text area before the URL is run.

**InvalidMessage** is the message that will be displayed in an alert pop-up if the user does not enter valid values into the text area.

#### Example:

An example of a WETreePicker function with parameter values entered would be

```
stringvar items:= "Product|Product Line|Product Category|Product
                  Name|Geography|Region|Country";
stringvar hierarchy:= '0|1|1|1|0|1|1';
stringvar picks:= 'n|y|y|y|n|y|y';
```

```
WETreePicker("groups", items, hierarchy, picks, join({?groups},"|"), "3cm", "4cm", "", "",
"empty", "Please enter at least one value to group on")
```

### WETreePickerExt(ElementName, ElementValues, ElementDisplays, ElementHierarchy, ElementPicks, ElementDefault, TextAreaHeight, TextAreaWidth, TextAreaFont, ElementLabels, Validation, InvalidMessage)

The WETreePickerExt function is used to create a tree picker control in a report. The tree menu can have multiple levels of hierarchy. Values picked from the tree are inserted into a text area.

The WETreePickerExt function differs from the WETreePicker function in that the extended (Ext) function passes a value that can differ from the display. For example a customer name can be displayed while a customer ID is passed from the control.

The WETreePickerExt function syntax can be edited to create a desired look and feel for your tree menu. Menu fonts and icons can be customized within the function syntax. Open the WETreePickerExt in **Report > Formula Workshop** for more information.

#### Parameters:

**ElementName** must be set to the name of the prompt that the tree picker control satisfies. For example, if the prompt is {?groups}, then the ElementName parameter must be set to "groups".

**ElementValues** is a character separated array of what will be passed to a resultant URL based on the chosen values in the tree menu. Values can include text objects (using quoted text or WETextObject functions) for folders and values.

Separate multiple values with the "|" character.

**ElementDisplays** is a character separated array of what will be displayed in the tree menu. Values can include text objects (using quoted text or WETextObject functions) for folders and values.

Separate multiple values with the "|" character.

**ElementHierarchy** is a character separated array representing the hierarchy of the ElementValues of the tree menu. A zero (0) represents a parent or root folder level item where a one (1) represents the second level of hierarchy and a two (2) represents the third level of hierarchy.

Separate multiple values with the "|" character.

**ElementPicks** is a character separated array representing the values of the tree menu that are available to be chosen by the end user. A yes (y) represents a value that can be added into the chosen value text area. Element picks can be either a folder or child item. A no (n) represents a folder or value in the tree that cannot be added into the chosen value text area.

Separate multiple values with the "|" character.

**ElementDefault** is the value or values that you wish to display in the text area when the report is opened.

Separate multiple values with the "|" character.

If the prompt is used as the default, e.g. `split({?Country},"|")`, the values entered will appear in the text box when the target report is opened.

**TextAreaHeight** is a text value that can be in inches (e.g. "3in") centimeters (e.g. "6cm"), or pixels (e.g. "1600pi").

**TextAreaWidth** is a text value that can be in inches (e.g. "3in") centimeters (e.g. "6cm"), or pixels (e.g. "1600pi").

**TextAreaFont** is used to format the text in the text area. The report developer can use the `WEFont` function in the Formatting functions to easily define a formatted font style.

**ElementLabels** is a character separated value used to define the "Reset" and "Clear" label links at the bottom of the text area. For example "Reset Choices|Clear Choices".

If this parameter is left blank then the defaults of Reset and Clear are used.

**Validation** is used to ensure that the user enters a value into the text area before the URL is run.

**InvalidMessage** is the message that will be displayed in an alert pop-up if the user does not enter valid values into the text area.

**Example:**

An example of a `WETreePickerExt` function with parameter values entered would be

```
stringvar values:= "P|P1|P2|P3|G|G1|G2";
stringvar displays:= "Product|Product Line|Product Category|Product
                    Name|Geography|Region|Country";
stringvar hierarchy:= '0|1|1|1|0|1|1';
stringvar picks:= 'n|y|y|y|n|y|y';
WETreePickerExt ("groups", values, displays, hierarchy, picks, join({"?groups"},"|"), "3cm",
                "4cm", "", "empty", "Please enter at least one value to group on")
```

## Menus

This section includes functions in the Menus folder of the webElements function library.

### WETreeMenu(ElementName, ElementValues, ElementHierarchy)

The WETreeMenu function is used to create a tree menu in a report. The tree menu can have multiple levels of hierarchy.

The WETreeMenu function syntax can be edited to create a desired look and feel for your tree menu. Menu fonts and icons can be customized within the function syntax. Open the WETreeMenu in **Report > Formula Workshop** for more information.

#### Parameters:

**ElementName** is a unique name that the report developer assigns to the tree menu.

**ElementValues** is a character separated array of what will be displayed in the tree menu. Values can include text objects (using quoted text or WETextObject functions) for folders and links to open content (using WEOpenInTargetLink or WEOpenInNewWindowLink).

Separate multiple values with the “|” character.

**ElementHierarchy** is a character separated array representing the hierarchy of the ElementValues of the tree menu. A zero (0) represents a parent or root folder level item where a one (1) represents the second level of hierarchy and a two (2) represents the third level of hierarchy.

Separate multiple values with the “|” character.

#### Example:

An example of a WETreeMenu function with parameter values entered would be

```
Stringvar menuvalues:= "Sales" + "|" + {@saleslink1} + "|" + {@saleslink2} + "|" + "HR" + "|" +  
    {@hrlink1};  
Stringvar menuhier:= "0|1|1|0|1";  
Stringvar menu1:= WETabMenu("menu1", menuvalues, menuhier);
```

## Other

This section includes functions in the Other folder of the webElements function library.

### ArrayPositionFinder(SearchIn, SearchFor)

The ArrayPositionFinder function is used to find the position of a specific value in a String Array. This function is used internally by several of the webElements functions.

**Parameters:**

**SearchIn** is the String Array that is being searched.

**SearchFor** is the String that is to be found in the Array defined in the SearchIn parameter.

Printing or Exporting Reports that use webElements

**Formulae containing webElements print as text or export as text.**

This behaviour is as per the support policy for pass-through html. Formulae containing pass-through html will print as text and not as rendered html objects.

The following sections will give you ideas on how to work around this issue.

**Use the browser print feature should you wish to print the controls.**

A workaround if you wish to print the controls as seen in the browser is to use the browser print feature instead of the viewer print feature.

**Format the formula to hide the webElements text when printing or exporting.**

A workaround should you not wish to print the controls is to format the formula on the report to have white text and to use the font formatting within the function itself to format the object while in the viewer. Upon printing using the viewer print feature, the formula font will be white and will not show up. This may be a suitable workaround for some export types as well.

**Use an IFrame (WEIFrame function) to contain the information that you wish to print or export.**

A workaround should you wish to print the contents of a report, including charts, but do not wish to print the controls is to use an IFrame for the main report contents.

1. Create a “container” report that only has the webElements controls (select menus, text boxes, submit buttons, etc.) in it.
2. Create a new formula using the WEIFrame (in the Formatting folder) and insert this new formula on your “container” report.
3. Have the IFrame formula point to the report that contains your content / data.
4. On the container report, create a new formula using the WEViewer function and choose Suppress for the toolbar parameter so that you don't see a toolbar on the container report.
5. Optional: On the main content / data report you can hide /show the toolbar using the WEViewer function and choosing Hide as the toolbar parameter and Hide as the Scrollbar parameter...now the user can print from the content report instead of the report with the controls...hence none of the controls will print or export as the IFrame drives this content.
6. Optional: In the content / data report format the formula that contains the WEViewer function as white...when it prints it will not show up. It will still export to some formats such as Excel, but the number of controls that do export will be minimized.

## Troubleshooting

**All of the formulae containing inputs and controls show up as text in the SAP BI / BusinessObjects HTML viewers.**

Ensure that an unsuppressed WEBUILDER function appears in your report. This function should contain all of the webElements that interact with the target report.

**Formulae containing webElements print as text or export as text.**

This behaviour is as per the support policy for pass-through html. Formulae containing pass-through html will print as text and not as rendered html objects.

See the Printing or Exporting Reports with webElements section for more information and workarounds.

**All of the inputs and controls show up correctly on the report; but, nothing is passed to the target report when you click the submit button.**

Ensure that an unsuppressed WEBUILDER function appears in your report. This function should contain all of the webElements that interact with the target report.

**All of the inputs and controls show up on the report; however, the report retains its original parameter values when you click the submit button. No error appears at the left side of the browser's Status Toolbar.**

This problem occurs when data is saved with the report.

To ensure that data is not saved with the report, select **File > Report Options** and clear the **Save Data With Report** checkbox.

**All of the inputs and controls show up on the report; but, "Error on page." appears at the left side of the browser's Status Toolbar when a submit button or submit link is clicked. Double-clicking on the error icon reveals that a "'getform' is undefined" error has occurred.**

This problem can occur when there is no formula containing a WEBUILDER function on the report, or the formula containing the WEBUILDER function is suppressed or on a different page than the other webElements controls.

See the *Requirements for all Crystal Reports* section of this user guide for more information.

**All of the inputs and controls show up on the report; but, "Error on page." appears at the left side of the browser's Status Toolbar when a submit button or submit link is clicked. Double-clicking on the error icon reveals that a "'getform' is undefined" error has occurred.**

This problem can occur when a prompt on the report and the corresponding ElementName for the control starts with a number or contains non-alphanumeric characters.

The ElementNames are used for JavaScript variables for storing values. A variable name can consist of alphanumeric characters and the underscore. It cannot begin with a numeral. Variable names are case sensitive.

**In the report designer Preview mode no text shows up in the formula containing the WEBUILDER function.**

Verify that formulae and variables used in the formula containing the WEBUILDER function return a value. Null values will interfere with the proper results.

To disable Null values, select **File > Report Options** and check the **Convert Database NULL Values to Default** checkbox and the **Convert Other NULL Values to Default** checkbox.

**“Report Linking Error. Parameter parsing problem: String index out of range: -2” appears after clicking a submit button.**

A URL is being passed with null values for one or more IsS or IsM parameters. Ensure that you validate parameters that may return null values.

**“A string can be at most 65534 characters long.” appears in the designer when running a report containing webElements formulae.**

A formula in Crystal Reports can be no more than 64k in length.

Use one formula per webElements control to minimize the amount of data that is returned per formula.

If you are rolling up data in variables to use in webElements controls, use one formula per variable to minimize the amount of data that is returned per formula.

**Symbols on calendar controls and other controls do not show properly.**

Ensure that you have the Arial font installed on machines which view the reports. If this is not an option, change the icons and the appropriate icon fonts within the calendar controls to a symbol and font type that your end users will have installed.

**Controls not working properly in FireFox.**

Certain controls might not render properly in Mozilla FireFox. These include the marquee controls and the fly-out section control. Certain functionality is not supported in FireFox which prevents these controls from working properly in this browser.

**WESelectCascadeExt or WESelectMultiCascadeExt controls are not passing the correct values in the final URL.**

Ensure that in your cascade set that you do not use non-Ext controls in conjunction with the Ext controls. Non-Ext controls do not have separate values & displays and will cause issues with Ext controls.

**Select menu controls always appear on the top of any object regardless of the order they are placed in.**

Select lists do not obey the normal z-index stacking order. Nothing can be placed "on top of" a select list unless it is another select list with a higher z-index. Other elements are always rendered below select lists, even if they are given a higher z-index value than the select.

As a workaround place the select menu into a collapsible section or a pull-down section or a fly-out section. Otherwise ensure there is enough space around select menus to avoid this issue.



## Tips and Tricks

**A report containing prompts and webElements controls is published to the Enterprise environment. How can the default prompt page be avoided?**

Move the report into a folder that users will not access. In the **Central Management Console > Objects > New Object** interface create a **Hyperlink** that will open the report with chosen default prompt values. Publish the hyperlink to the folder that the users will access.

**Example:**

The following is an example hyperlink for BI 4.3 (Crystal Reports 2020) using OpenDocument syntax to open a Crystal Report with default prompt values.

```
/BOE/OpenDocument/opendoc/openDocument.jsp?  
sType=rpt&slDType=CUID&iDocID=AaWLhmtS9kVHvRSSmRzDjk8&lsSCustomer=6&lsCou  
ntry=Canada
```

**Can webElements be used with live data?**

Yes. Controls such as select menus that require that data be rolled up or amalgamated before the control is created are populated by running a Report Header sub-report that rolls the data up. The data is then passed to the main report via a shared variable which is used in the control.

Please see the DCP report in the webElements .zip package for ideas on how to do this.

Controls such as check boxes or radio buttons can be populated by placing the formula containing the control in the Details section or the Group Header / Footer section where the control is needed. Ensure that the controls do not span more than one page. See the Requirements section of the User Guide for more information.

**When is a WEBUILDER function required on a report?**

A WEBUILDER function is required on a report when webElements controls (e.g. WESelect, WETable, WETextBox, etc.) are used on the report. WEBUILDER creates JavaScript functions and variables that are used to build a URL or to pass control values to another viewer.

## Finding more information

For more information and resources related to webElements, visit the Crystal Reports design forum at <http://forums.sdn.sap.com/forum.jspa?forumID=300&start=0>

When posting to the forums, ensure that you use “webElements” in your subject line.