How To...
Integrate SAP NetWeaver 2004s BI Application UI Elements and Visual Composer UI Elements by service-enabling UI Elements
Version 4.00 – November 2006

Applicable Releases:
SAP NetWeaver 2004s
Index

1. Scenario

2. Version History

3. Introduction

4. Data Provider Information

5. Step by Step Solution
   4.1 Build the Outbound Service Template Web Application
   4.2 Build the Inbound Service Template Web Application

6. Scenarios
   5.1 Hierarchy Dialog within VC
   5.2 Hierarchical Filter within VC
   5.3 Listbox within VC using 2 separate iViews.
   5.4 Use Visual Composer Slider Element to update BI Web Application

7. Guidelines

8. Troubleshooting

9. Appendix
1 Scenario

In this scenario, we will integrate UI Elements from BI Web Application Designer such as hierarchies, listboxes, and dialogs with UI Elements from Visual Composer such as tables and other UI elements. Most of the scenarios are built off SAP DEMO content just to show examples.

2 Version History

Version 3.0
- Originally released for customers

Version 4.0
- Added support for master data values that have spaces
- The Outbound script did not send the code in the correct form for elements that are not <CHARACTERISTICS>. This has been corrected.
- Added support for objects in customer namespaces.
- Added clarification for webapi commands for different hierarchy types
- Added troubleshooting section
- Changed "inbound" logic to skip filtering of null values passed from VC
- Added support for variables of type characteristic. Previously, only formula variables were supported.

3 Introduction

The paper will show how to integrate UI Elements between Visual Composer and Web Application Designer. This concept can be extended to any SAP toolset, such as Webdynpro for ABAP, BSPs, WebDynpro for Java, etc…

The goal of this solution is to make every UI element a service. This service will utilize portal eventing capabilities to send and receive XML documents to integrate these applications.

In this scenario, we will create a service out template and a service in template which will send and receive XML documents in this format:
<Params version="2"><Row F0CALMONTH="200407"/> <Row F0CALMONTH="200408"/></Params>
4 Overview of “Data Provider – Information” Web Item

The Data Provider – Information Web Item returns an XML tree in your web application. This xml tree isn’t displayed in the web application but is available in the source code for use. This paper makes use of this XML Tree.

This xml tree has the following format:

```xml
<xml id="DATA_PROVIDER_INFO_ITEM_1">
   <BICS_VIEW>
      <RESULT_SET>
         <AXES>
            <AXIS name="ROWS">
                ...
            </AXIS>
            <AXIS name="COLUMNS">
                ...
            </AXIS>
         </AXES>
         <DATA_CELLS>
             ...
         </DATA_CELLS>
      </RESULT_SET>
   </BICS_VIEW>
</xml>
```

This is only a subset of the tree. Check the source of your web applications for an example of a full tree. This solution is primarily driven off this XML dataset. An XSL transform is converting this XML format to a format understood by Visual Composer.

5 The Step By Step Solution

5.1 Build the Outbound Service Template Web Application

In this step, we will build a web template that has our logic to send out the selection state of your web application as an xml document. This template can be embedded as a nested web template in any web application. This is a one time event and typically should be done by someone from IT or development at customer sites. The web application designer end user won’t have to code to build their template. They will just use this service template and nest it in their applications!
1. Login to the SAP NetWeaver 2004s BI Web Application Designer

2. Create a new Web Template.

3. Double Click on “New Data Provider” to create a new data provider.
4. Specify DP_1 as the data provider and leave the query, query view, and InfoProvider fields blank. Choose OK.

5. Add the “Data Provider – Information” web item to the web application specify the web item parameters as “Output Navigation Status=On” and “Output Result Data=Off”

6. Drag and drop the “Script” Web Item into your web application.
7. Choose the help selector for “Script” in the “SCRIPT_ITEM_1” web item.

8. Add the scripting code specified in the Appendix.

9. Go to the XHTML tab and ensure the script loads in the body:
   
   <script type="text/javascript"
   language="JavaScript">loadIt();</script>

   See the appendix for full XHTML source code.
10. Save the web template.

11. Name the web template “Z_SERVICEOUTBOUND” with description “Outbound Service Template”

5.2 Build the Inbound Service Template Web Application

In this step, we will build a web template that has our logic to receive variable values or filter values as an xml document. If you need to cover other inbound items other than filter values and variable values, you can update this inbound service template. This template can be embedded as a nested web template in any web application. This is a one time event and typically should be done by someone from IT or development at customer sites. The web application designer end user won’t have to code to build their template. They will just use this service template and nest it in their applications!
1. Login to the SAP NetWeaver 2004s BI Web Application Designer

2. Create a new Web Template.

3. Double Click on “New Data Provider” to create a new data provider.
4. Specify DP_1 as the data provider and leave the query, query view, and InfoProvider fields blank. Choose OK.

5. Drag and drop the “Script” Web Item into your web application.

6. Choose the help selector for “Script” in the “SCRIPT_ITEM_1” web item.
7. Add the scripting code specified in the Appendix for the inbound service web template.

8. Save the web template.

9. Add the Data Provider – Information Web Item to your Web Template.
10. Name the web template “Z_SERVICEINBOUND” with description “Inbound Service Template”

6 Scenarios to use Service Template

6.1 Hierarchy Dialog within VC

In this scenario, we want to use our service template we created to integrate a hierarchy popup filter dialog within a Visual Composer Application.

The output of the scenario as follows:

The query to use for this example is "0D_DX_M01_Q0003" as this includes a hierarchy. You can use your own query as well.

The webAPI sequence we will use to submit the hierarchy filter to BI from Visual Composer is listed below. Keep in mind that the API command is different for hierarchies based with nodes that are characteristic values versus nodes that are based off 0HIER_NODE
WebAPI for Hierarchy based off characteristic values:

```
WEBAPI = 'FILTER_IOBJNM=0D_DBSIC1;FILTER_NODE_IOBJNM=0D_DBSIC1;FILTER_VALUE=' & @F0D_DBSIC1
```

WebAPI for Hierarchy based off 0HIER_NODE:

```
WEBAPI = 'FILTER_IOBJNM=0COUNTRY;FILTER_NODE_IOBJNM=0HIER_NODE;FILTER_VALUE=' & @F0COUNTRY
```

NOTE: This web api is deprecated with the next release of SAP NetWeaver and is subject to change.

1. Login to the SAP NetWeaver 2004s BI Web Application Designer
2. Create a new Web Template.

3. Insert a new data provider in your template.

4. Choose demo content query “0D_DX_M01_Q0003” or your own query that you want to send out values from.
5. Drag and drop the “Button Group” web item into your web application.

6. Choose the help selector to create a button.

7. Specify caption “Industry Value Help” and quick info “Industry Value Help”. Choose the help selector for the command.
8. Choose the "OPEN_SELECTOR_DIALOG" command and choose Next.

9. Choose Data Provider Affected as "DP_1" and choose Next.
10. Choose characteristic “0D_DBSIC1” and choose OK.

11. Choose OK to finish the creation of the button.

12. Drag and drop the “Web Template” item into your web application.
13. Specify your service web template “Z_SERVICEOUTBOUND” as the nested web template.

14. Save your web application.

15. Save your web template with description “Hierarchy Button” and technical name “Z_HIER_BUTTON”.
16. Make sure you save this URL as you'll need it in your Visual Composer model.


Create a new model.

18. Name your model.
19. Drag and drop an iView into your storyboard. Double click on the iView to enter the iView.

20. Go to Find Data and drag and drop

21. Drag and drop a table view from the output port.
22. Choose the following fields and click OK.

23. Drag and drop a start point from the input port of the dataservice.

24. Choose OK.
25. Drag and drop a “Signal In” element into your storyboard.

26. Rename the “in1” event to “Filter”.

27. Add a field to your “Filter” event.
28. Name the field 0D_DBSIC1 and choose OK.

29. Connect the "Filter" event to the input port.

30. Specify a formula for the web api on the line connecting the "Filter" event into the data service.
31. Specify the following mapping for the web api and choose OK.

   NOTE: This web api is deprecated with the next release of SAP NetWeaver and is subject to change.

32. Drag and drop an HTML View into your model and name it “Hierarchy”.

33. Connect the HTML View to the start point.
34. Specify a formula for the URL.

35. Enter the formula for your web application which has the hierarchy button.

36. For the HTML View, set the display as transparent.
37. Go to the layout tab and update the layout such that the button is displayed at the top of the page.

38. Deploy your model.
39. Run your model. You now have a hierarchy value help available for VC filtering...

6.2 Hierarchical Filter within VC

In this scenario, we want to use our service template we created to integrate a hierarchical filter web item into a Visual Composer Application.

**WebAPI for Hierarchy based off characteristic values:**

<table>
<thead>
<tr>
<th>BRANCHE</th>
<th>SIC_HIERARCHIE</th>
<th>InfoObject</th>
<th>Node Na</th>
<th>Li</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SIC Hierarchy</td>
<td>0HIER_NODE 0D_DBSIC1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Agriculture, forestry and fishing</td>
<td>0D_DBSIC1</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wheat farm</td>
<td>0D_DBSIC1</td>
<td>0111</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rice farm</td>
<td>0D_DBSIC1</td>
<td>0112</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Corn farm</td>
<td>0D_DBSIC1</td>
<td>0115</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Soybean farm</td>
<td>0D_DBSIC1</td>
<td>0116</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cash grains farm</td>
<td>0D_DBSIC1</td>
<td>0119</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cotton farm</td>
<td>0D_DBSIC1</td>
<td>0131</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tobacco farm</td>
<td>0D_DBSIC1</td>
<td>0132</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sugarcane/sugar beet farm</td>
<td>0D_DBSIC1</td>
<td>0133</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Irish potato farm</td>
<td>0D_DBSIC1</td>
<td>0134</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Field crop farm</td>
<td>0D_DBSIC1</td>
<td>0138</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vegetable/melon farm</td>
<td>0D_DBSIC1</td>
<td>0161</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Berry crop farm</td>
<td>0D_DBSIC1</td>
<td>0171</td>
<td></td>
</tr>
</tbody>
</table>

```
WEBAPI = 'FILTER_IOBJNM=0D_DBSIC1;FILTER_NODE_IOBJNM=0D_DBSIC1;FILTER_VALUE=' @F0D_DBSIC1
```

**WebAPI for Hierarchy based off 0HIER_NODE:**

<table>
<thead>
<tr>
<th>Country Hierarchy</th>
<th>InfoObject</th>
<th>Node N</th>
<th>Li</th>
</tr>
</thead>
<tbody>
<tr>
<td>World</td>
<td>0HIER_NODE WORLD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Africa</td>
<td>0HIER_NODE AFRICA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nigeria</td>
<td>0COUNTRY NG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Europe</td>
<td>0HIER_NODE EUROPE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sweden</td>
<td>0COUNTRY SE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Poland</td>
<td>0COUNTRY PL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Netherlands</td>
<td>0COUNTRY NL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finland</td>
<td>0COUNTRY FI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Switzerland</td>
<td>0COUNTRY CH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
WEBAPI = 'FILTER_IOBJNM=0COUNTRY;FILTER_NODE_IOBJNM=0HIER_NODE;FILTER_VALUE=' & @F0COUNTRY

NOTE: This web api is deprecated with the next release of SAP NetWeaver and is subject to change.

The output of this scenario looks as follows:

1. Login to the SAP NetWeaver 2004s BI Web Application Designer
2. Create a new Web Template.

3. Insert a new data provider in your template.

4. Choose demo content query “0D_DX_M01_Q0003” or your own query that you want to send out values from.
5. Drag and drop the "Hierarchical Filter" web item into your web application.

6. Set the characteristic "0D_DBSIC1" as the source for the hierarchy.
7. Drag and drop the “Web Template” Item into your web application.

8. Specify your service web template “Z_SERVICEOUTBOUND” as the nested web template.

9. Save your web application.
10. Save your web template with description “Hierarchy FILTER” and technical name “Z_HIER_FILTER”.

11. Make sure you save this URL as you’ll need it in your Visual Composer model.

Create a new model.
13. Name your model.

14. Drag and drop an iView into your storyboard. Double click on the iView to enter the iView.

15. Go to Find Data and drag and drop
16. Drag and drop a table view from the output port.

17. Choose the following fields and click OK.

18. Drag and drop a start point from the input port of the dataservice.
19. Choose OK.

20. Drag and drop a “Signal In” element into your storyboard.

21. Rename the “in1” event to “Filter”.
22. Add a field to your "Filter" event.

23. Name the field 0D_DBSIC1 and choose OK.
24. Connect the “Filter” event to the input port.

25. Specify a formula for the web api on the line connecting the “Filter” event into the data service.

26. Specify the following mapping for the web api and choose OK.

NOTE: This web api is deprecated with the next release of SAP NetWeaver and is subject to change.
27. Drag and drop an HTML View into your model and name it "Hierarchy".

28. Connect the HTML View to the start point.

29. Specify a formula for the URL.
30. Enter the formula for your web application which has the hierarchy button.

![Assign Value window](image1)

31. For the HTML View, set the display as transparent.

![HTML View](image2)

32. Go to the layout tab and update the layout such that the button is displayed at the top of the page.

![Layout Tab](image3)
33. Deploy your model.

34. Run your model. You now have hierarchy filtering available for VC filtering...

6.3 Listbox within VC using 2 separate iViews.

In this scenario, we want to use our service template we created to integrate a listbox web item with a Visual Composer Application that are located in 2 separate Portal iViews.

The output of this scenario looks as follows:
1. Launch the Web Application Designer.

2. Create a new web application.

3. Double Click on "New Data Provider" to create a new data provider.
4. Specify query “0D_DX_M01_Q0006” for DP1 and choose OK.

5. Drag and drop the “Listbox” web item into your web application.

6. Go to the Web Item Parameters of the “LISTBOX_ITEM_1” web item and specify the “Affected Data Provider” as “DP1”, and the “Characteristic” as “0CALMONTH”. Also, make sure “Label Visible” is checked “On”
7. Drag and drop the “Web Template” Item into your web application.

8. Specify your service web template “Z_SERVICEOUTBOUND” as the nested web template.

9. Save the web application
10. Save with description “GR00 Send WAD Listbox to VC” and technical name “Z_GR00_LISTBOXOUT”.

11. Choose “Web Template -> Publish -> to Portal”

12. Choose the help selector for the “Folder Name”
13. Choose “Portal Content/Workshop/GR00”. If this folder doesn’t exist, create it in the PCD.

14. Choose “Execute” to save this in the PCD.

15. Login to Visual Composer at http://<host>:<port>/VC/
16. Choose “Model -> New Model”

17. Specify the name “GR00_RECV_FROM_WAD_LISTBOX”

18. Add an iView to the model and name is “GR00 Recv from WAD Listbox” where ## is your group number.
19. Double click on the iView to enter the contents of this iView.

20. Choose “Find Data”

21. Choose your BW System and query “0D_DX_M01_Q0006”.
22. Drag and drop this query into your iView.

23. From the input port, drag and drop out a “Singal In”

24. Choose OK.
25. Rename the “in1” object to “Filter”

26. Choose “Add” to add a field to your filter.
27. Specify Field name “0CALMONTH”

28. Drag and drop a start point from the Input port.

29. Choose OK.
30. Click on the line “Filter” line to the input port and map “Calendar Year Month” to “F0CALMONTH”

31. Drag and drop a “table” from the output port.

32. Choose the following fields and click OK.
33. Go to deploy.

34. Choose Deploy.

35. Go to the portal.
36. Navigate to “Content Administration - > Portal Content -> Visual Composer -> Models -> GR00_RECV_FROM_WAD_LISTBOX -> iviews -> GR00 Recv From WAD Listbox” and right click and choose “Copy”.

37. Navigate to “Portal Content -> Workshop -> GR00” and right click on this folder and choose “Paste as Delta Link”.

38. Right click on folder “Portal Content/Workshop/GR00” and create a new page.
39. Enter page name “GR00 Listbox WAD to VC” and page ID “GR00_LBOX_WAD_TO_VC” and choose next.

40. Choose the “Default Page Template” and click Next.

41. Choose “2 Columns (Narrow:Wide)” and click Add.
42. Click Next.

43. Choose Finish.

44. Choose “Open the object for editing” and click OK.
45. Right click on iView “GR00 Send WAD Listbox to VC” and click “Open -> Object”

46. Choose “Appearance -> Size”

47. Specify the following parameters and save.
48. Close this iView

49. Right click on “GR00 Send WAD Listbox to VC” and choose “Add iView to Page -> Delta Link”

50. Right click on “GR00 Recv from WAD Listbox” and choose “Add iView to Page -> Delta Link”
51. Go to the page layout

52. Ensure the layout is specified as below and save the page.

53. Choose Preview.
54. You should now be able to apply filters from the WAD Listbox that will affect VC.

6.4 Use Visual Composer Slider Element to update BI Web Application

In this scenario, we want to use our service template we created to integrate a slider from Visual Composer with an analysis web item in a BI Web Application. The example here will be updating a variable value that is set from a VC slider and will update a filter value with Calendar Year/Month.

The output of this scenario looks as follows:
1. Login to the SAP NetWeaver 2004s BI Web Application Designer

2. Create a new Web Template.

3. Insert a new data provider in your template.
4. Choose a query to use as your data provider. In this case, we are using a copy of 0D_DX_M01_Q0006 which has a Top N Condition on Net Sales. The Top N Condition is filled by variable Z_ENTRYTOP which is a user entry, mandatory variable with default value 10.

5. Drag and drop the "Analysis" web item into your web application.

6. Drag and drop the "Web Template" Item into your web application.
7. Specify your service web template “Z_SERVICEINBOUND” as the nested web template.

8. Save your web application.

9. Save your web template with description “Analysis Inbound” and technical name “Z_ANALYSIS_IN”.
10. Make sure you save this URL as you'll need it in your Visual Composer model.

Create a new model.

12. Name your model.
13. Name the model “Slider_Integration”

14. Drag and drop an iView in your model and name it “Slider_Integration”

15. Double click on the “Slider_Integration” iView.
16. Drag and drop an HTML View in your model and name it “Analysis”.

17. Drag and drop a Start Point from the in port on the HTML View.

18. Choose “url” and click OK.
19. Double click on the line connecting the start point and the in port.

20. Double click on the url field to enter a formula.

21. Paste the URL for your BI Web Application and click OK.
22. Drag and drop a “Form view” element into your model and name it “Filter”.

23. Go to the Field area of the form and click Add.

24. Choose the control type “HSlider” and name the field "Z_ENTRYTOP."
25. Choose "Properties" for your field "Z_ENTRYTOP".

26. Specify the following Display.

27. Specify the following properties for Range.
28. Specify the custom action "Variable" and choose Close.

29. Drag and drop a “Signal Out” element from the out port of the Filter Form.

30. Choose the field “Z_ENTRYTOP” and click OK.
31. Double click on the line connecting the form and output signal and name this event “Variable”.

32. Double click on the output signal and name the output signal “Variable”.

33. Double click on your Filter Form.
34. Add a Field to your form.

35. Choose "Drop-down List" and specify Field "0CALMONTH".
36. Go to the properties of your “F0CALMONTH” field.

37. Specify the following properties for the “Display” tab.

38. Choose “Dynamic” for the Entry List.
39. Choose a query with “0CALMONTH” in it and choose this field.

40. Specify the following properties for the “Entry List” Tab.
41. Specify Action “Filter” and click Close.

42. Drag and drop a “Signal Out” from the out port of the Filter.
43. Choose field “FOCALMONTH” and click OK.

44. Double click on the line between the out port and the signal out and name the event “Filter”.

45. Double click on the signal out and name the output “Filter”.

46. Go to the Layout and redraw the layout as displayed here:

47. Save the model.

48. Deploy the model.
49. Run the model. The slider will set the top N Condition for Net Sales. The dropdown for Cal Year/Month will filter to an individual Calendar Year/Month.

7 Guidelines for Solution

For these solutions to work, you must follow these guidelines:

- Always name all filter fields within Visual Composer based off the InfoObject technical name (must be capitalized as fields is case sensitive)
- Always name all variable fields within Visual Composer based off the Variable technical name (must be capitalized as fields are case sensitive)
- Always name the signal in for a Visual Composer model “Filter”
- Always name the signal out for fields to filter as “Filter”
- Always name the signal out for variable values to set as “Variable”

8 Troubleshooting

8.1 Flash Debugger

Visual Composer allows you to use the flash debugger. This will expose all actions and events within an individual Visual Composer application.
1. Within Visual Composer, go to Tools -> Options.

2. Go to the Compiler tab. Choose the “Add Flash debugging console” option and choose OK.

3. Deploy your Visual Composer application.
4. Run your VC Application. You will now see the Flash Debug Logger which will show you all events taking place at runtime!!

8.2 Alerts within Javascript

Because we are working with multiple applications, as we broadcast or send out events, to ensure this occurs properly, you can use alerts within javascript

1. Within your Z_SERVICEOUTBOUND web template, you can add an alert to show the parameters you are passing out to Visual Composer.

```javascript
function loadIt()
{
    DATA_PROVIDER_INFO_ITEM_1.async = false;
    var sendout;
    sendout = DATA_PROVIDER_INFO_ITEM_1.firstChild.childNodes[5].transformNode(xsl_transform_0.documentElement);
    // To debug, comment the line below back in.
    // alert(sendout);
    EPCMPROXY.raiseEvent( "urn:com.sap.vc:epcm", "Filter", sendout , null );
}
</script>
```
2. Within your Z_SERVICEINBOUND web template, you can add an alert to show the parameters you are receiving from Visual Composer.

Comment back in the alert(unesc_val.xml) to view the object.

```javascript
function setSequence(action, eventObj)
{
    //Create a new object of type sapbi_CommandSequence
    var commandSequence = new sapbi_CommandSequence();

    xml_in = eventObj.dataObject;
    unesc_val = unescape(xml_in);
    // alert(unesc_val.xml);
}
```

8.3 Common Errors – EPCMPROXY undefined

Ensure you use the fully qualified domain name when accessing your portal and when asking Visual Composer. Also, ensure that the fully qualified domain name is the same between VC and WAD.

8.4 HttpWatch

HttpWatch allows you to track html calls. This requires that this software is installed. Also, keep in mind that this doesn’t offer you internal commands to flash and will help with eventing between portal iViews, but not off much assistance within the flash runtime. The Flash Debugger should be used for the flash runtime. HttpWatch will help immensely for web application designer dashboards.

1. Run your web application and turn on the HTTPWATCH with the icon on your toolbar.
2. Choose Start to log http calls.

3. Trigger an event that will update other iViews.

4. All URL calls are logged and you can use this to ensure that filter values are passed properly...

   Also, ensure epfcproxy.js is loaded when the iviews load.
9 Appendix

9.1 Source Code for Script Web Item in Service Out Template:

```javascript
function loadIt() {
    DATA_PROVIDER_INFO_ITEM_1.async = false;
    var sendout;
    sendout = DATA_PROVIDER_INFO_ITEM_1.firstChild.childNodes[5].transformNode(xsl_transform_0.documentElement);
    // To debug, comment the line below back in.
    // alert(sendout);
    EPCM_PROXY.raiseEvent( "urn:com.sap.vc:epcm", "Filter", sendout, null );
}
```

## XSL Transformation that displays Filter Values
```
<xml id="xsl_transform_0">
    <xsl:template match="text() | @* ">
        <!-- Skip output of tags, which are not CHARACTERISTICS -->
    </xsl:template>

    <xsl:template match="CHARACTERISTICS">
        <xsl:element name="Params">
            <xsl:attribute name="version">2</xsl:attribute>
            <xsl:for-each select="//CHARACTERISTIC">
                <xsl:variable name="var1">
                    <!-- Attributname for row-tag is created here -->
                    <xsl:choose>
                        <xsl:when test="starts-with(@name,'0')"> <!-- A number is not allowed as first char of a xml element, so a F is concatenated -->
                            <xsl:value-of select="concat('F',@name)"/>
                        </xsl:when>
                        <xsl:when test="contains(@name,'/')"> <!-- Custom namespace compatibility: A / is not allowed as first char of a xml element, so a F is concatenated and slashes are transformed into underscores -->
                            <xsl:value-of select="concat('F', translate(@name, '/', '_'))"/>
                        </xsl:when>
                        <xsl:otherwise> <!-- All other elements are fine, just take their name -->
                            <xsl:value-of select="@name"/>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:variable>
                <xsl:for-each select="SELECTIONS/SELECTION">
                    <xsl:variable name="var2">
                        <!-- Value of new attribute -->
                        <xsl:choose>
                            <xsl:when test="@type = 'SINGLE_MEMBER'"> <!-- Value of new attribute -->
                                <xsl:value-of select="MEMBER/@name"/>
                            </xsl:when>
                            <xsl:otherwise> <!-- Value of new attribute -->
                                <xsl:value-of select="VALUE/MEMBER/@name"/>
                            </xsl:otherwise>
                        </xsl:choose>
                    </xsl:variable>
                    <xsl:element name="Row">
                        <!-- Creation of row-tag -->
                        <xsl:attribute name="{$var1}">
                            <!-- Adding attribute with the name of var1 -->
                            <xsl:value-of select="$var2"/>
                        </xsl:attribute>
                    </xsl:element>
                </xsl:for-each>
            </xsl:for-each>
        </xsl:element>
    </xsl:template>

    <xsl:apply-templates/>
```
```
9.2 Source Code for Service Out Web Template:

```xml
  <html>
    <head>
      <title>Netweaver BI Web Application</title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    </head>
    <body>
      <bi:QUERY_VIEW_DATA_PROVIDER name="DP_1">
        <bi:INITIAL_STATE type="CHOICE" value="QUERY" />
        <bi:QUERY value="" />
      </bi:INITIAL_STATE>
      <bi:TEMPLATE_PARAMETERS name="TEMPLATE_PARAMETERS" />
      <!-- insert data providers, items and other template content here -->
      <bi:DATA_PROVIDER_INFO_ITEM name="DATA_PROVIDER_INFO_ITEM_1" designheight="120" designwidth="300" />
      <bi:DATA_PROVIDER_REF value="DP_1" />
      <bi:DATA_PROVIDER_INFO_ITEM>
        <bi:SCRIPT_ITEM name="SCRIPT_ITEM_1" designwidth="300" designheight="70">
          <bi:SCRIPT value="biLargeData:B0V1N4ARH3IKX18V29K7AT45Z" />
        </bi:SCRIPT_ITEM>
        <script type="text/javascript" language="JavaScript">loadIt();</script>
      </bi:DATA_PROVIDER_INFO_ITEM>
    </body>
  </html>
</bi:bisp>
```

9.3 Source Code for Script Web Item in Service In Template:

```javascript
function load(xmlFile) {
    // code for IE
    if (window.ActiveXObject) {
        xmlDoc = new ActiveXObject("Msxml2.DOMDocument.3.0");
        xmlDoc.loadXML(xmlFile);
    } else {
        alert("Your browser cannot handle this script");
    }
}

function loadVar(xmlFile) {
    // code for IE
    if (window.ActiveXObject) {
        vardoc = new ActiveXObject("Msxml2.DOMDocument.3.0");
        vardoc.loadXML(xmlFile);
    } else {
        alert("Your browser cannot handle this script");
    }
}
```
function getVarType(varname) {
    DATA_PROVIDER_INFO_ITEM_1.async = false;
    var sendout;
    sendout = DATA_PROVIDER_INFO_ITEM_1.firstChild.childNodes[3];
    loadVar(sendout.xml);
    var variables = vardoc.getElementsByTagName('VARIABLE');
    for(i=0; i < variables.length; ++i) {
        if(variables.item(i).getAttribute("name") == varname) {
            return variables.item(i).getAttribute("type");
        }
    }
}

function setVarSequence(eventObj) {
    setSequence('Variable', eventObj);
}

function setFilterSequence(eventObj) {
    setSequence('Filter', eventObj);
}

function setSequence(action, eventObj) {
    //Create a new object of type sapbi_CommandSequence
    var commandSequence = new sapbi_CommandSequence();

    xml_in = eventObj.dataObject;
    unesc_val = unescape(xml_in);
    // alert(unesc_val);
    load(unesc_val);

    // Get parameter version
    vers = xmldoc.getElementsByTagName("Params").item(0).getAttribute("version");
    if (vers == null) {
        vers = 0;
        alert('SP7 not supported');
    }

    // This is for SP8 and above
    if (vers == 2) {
        len_val = xmldoc.getElementsByTagName("Row").item(0).attributes.length;

        for(i = 0; i < len_val; ++i) {
            IOBJ = xmldoc.getElementsByTagName("Row").item(0).attributes[i].name;
            VAL = xmldoc.getElementsByTagName("Row").item(0).attributes[i].value;
            // Added this to support values with spaces...
            VAL = unescape(VAL);
            if(VAL == null || VAL == undefined || VAL == "") {
                // Don't filter on values that are null
            } else {
                //Strip of leading 'F' (primarily due to VC naming due to XML attributes not starting with "0")
                if (IOBJ.charAt(0) == 'F') {
                    IOBJ = IOBJ.substring(1, IOBJ.length);
                }
                if(action=='Variable') {
                    commandSequence = setVariableValue(commandSequence, IOBJ, VAL);
                }
                if(action=='Filter') {
                    commandSequence = setFilterValue(commandSequence, IOBJ, VAL);
                }
            }
        }
    }
}
function setFilterValue(commandSequence, IOBJ, VAL)
{
    //Create a new object of type sapbi_Command with the command named "SET_SELECTION_STATE_SIMPLE"
    var commandSET_SELECTION_STATE_SIMPLE_1 = new sapbi_Command("SET_SELECTION_STATE_SIMPLE");
    //Create parameter TARGET_DATA_PROVIDER_REF_LIST
    var paramTARGET_DATA_PROVIDER_REF_LIST = new sapbi_Parameter("TARGET_DATA_PROVIDER_REF_LIST", "");
    var paramListTARGET_DATA_PROVIDER_REF_LIST = new sapbi_ParameterList();
    //Create parameter TARGET_DATA_PROVIDER_REF
    var paramTARGET_DATA_PROVIDER_REF1 = new sapbi_Parameter("TARGET_DATA_PROVIDER_REF", "DP_1");
    paramListTARGET_DATA_PROVIDER_REF_LIST.addTabParameter( paramTARGET_DATA_PROVIDER_REF1 );
    //End parameter TARGET_DATA_PROVIDER_REF
    paramTARGET_DATA_PROVIDER_REF_LIST.setChildList( paramListTARGET_DATA_PROVIDER_REF_LIST );
    commandSET_SELECTION_STATE_SIMPLE_1.setTabParameter( paramTARGET_DATA_PROVIDER_REF_LIST );
    //End parameter TARGET_DATA_PROVIDER_REF_LIST!
    //Create parameter CHARACTERISTIC
    var paramCHARACTERISTIC = new sapbi_Parameter( "CHARACTERISTIC", IOBJ );
    commandSET_SELECTION_STATE_SIMPLE_1.addTabParameter( paramCHARACTERISTIC );
    //End parameter CHARACTERISTIC!
    //Create parameter RANGE_SELECTION_OPERATOR
    var paramRANGE_SELECTION_OPERATOR = new sapbi_Parameter("RANGE_SELECTION_OPERATOR", "EQUAL_SELECTION");
    var paramListRANGE_SELECTION_OPERATOR = new sapbi_ParameterList();
    //Create parameter EQUAL_SELECTION
    var paramEQUAL_SELECTION = new sapbi_Parameter("EQUAL_SELECTION", "MEMBER_NAME");
    var paramListEQUAL_SELECTION = new sapbi_ParameterList();
    //Create parameter MEMBER_NAME
    var paramMEMBER_NAME = new sapbi_Parameter("MEMBER_NAME", VAL);
    paramListEQUAL_SELECTION.addTabParameter( paramMEMBER_NAME );
    //End parameter MEMBER_NAME
    paramRANGE_SELECTION_OPERATOR.setChildList( paramListEQUAL_SELECTION );
    paramRANGE_SELECTION_OPERATOR.addTabParameter( paramRANGE_SELECTION_OPERATOR );
    //End parameter EQUAL_SELECTION!
    paramRANGE_SELECTION_OPERATOR.setChildList( paramListRANGE_SELECTION_OPERATOR );
    commandSET_SELECTION_STATE_SIMPLE_1.addTabParameter( paramRANGE_SELECTION_OPERATOR );
    //End command commandSET_SELECTION_STATE_SIMPLE_1
    return commandSequence;
}

function setVariableValue(commandSequence, VAR, VAL)
{
    // Define the Variable Type
    variablename = getVarType(VAR);
    if(variablename == 'DOUBLE')
    {
        variablename = 'FORMULA_VARIABLE';
    }
    if(variablename == 'INFO_OBJECT_MEMBER')
if(variabletype == null || variabletype == undefined || variabletype =="")
{
    variabletype = 'VARIABLE_INPUT_STRING';
}

//Create a new object of type sapbi_Command with the command named "SET_VARIABLES_STATE"
var commandSET_VARIABLES_STATE_1 = new sapbi_Command("SET_VARIABLES_STATE");

//Create parameter VARIABLE_VALUES
var paramVARIABLE_VALUES = new sapbi_Parameter("VARIABLE_VALUES", "" );
var paramListVARIABLE_VALUES = new sapbi_ParameterList();

//Create parameter VARIABLE_VALUE
var paramVARIABLE_VALUE1 = new sapbi_Parameter("VARIABLE_VALUE", "" );
var paramListVARIABLE_VALUE1 = new sapbi_ParameterList();

//Create parameter VARIABLE_TYPE
var paramVARIABLE_TYPE = new sapbi_Parameter("VARIABLE_TYPE", variabletype );
var paramListVARIABLE_TYPE = new sapbi_ParameterList();

//Create parameter FORMULA_VARIABLE
var paramFORMULA_VARIABLE = new sapbi_Parameter(variabletype , VAL );
paramListVARIABLE_TYPE.addParameter( paramFORMULA_VARIABLE );
//End parameter FORMULA_VARIABLE!

paramVARIABLE_TYPE.setChildList( paramListVARIABLE_TYPE );
paramListVARIABLE_VALUE1.addParameter( paramVARIABLE_TYPE );
//End parameter VARIABLE_TYPE!

//Create parameter VARIABLE
var paramVARIABLE = new sapbi_Parameter("VARIABLE", VAR );
paramListVARIABLE_VALUE1.addParameter( paramVARIABLE );
//End parameter VARIABLE!

paramVARIABLE_VALUE1.setChildList( paramListVARIABLE_VALUE1 );
paramVARIABLE_VALUES.addParameter( paramVARIABLE_VALUE1 );
//End parameter VARIABLE_VALUE!

paramVARIABLE_VALUES.setChildList( paramListVARIABLE_VALUES );
commandSET_VARIABLES_STATE_1.addParameter( paramVARIABLE_VALUES );
//End parameter VARIABLE_VALUES!

//End command commandSET_VARIABLES_STATE_1
//Add the command to the sequence
commandSequence.addCommand( commandSET_VARIABLES_STATE_1 );
return commandSequence;

9.4 Source Code for Service in Template

 xmlns:jsp='http://java.sun.com/JSP/Page'>
<title>Netweaver BI Web Application</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
   <!-- insert data providers, items and other template content here -->
</body>
</html>
9.5 References
