Fox Formula in SAP BI-Integrated Planning

Applies to:
This applies to SAP BI 7.X. For more information, visit the EDW homepage.

Summary
This document contains method of implementing a FOX (Formula extension) in SAP BW Integrated Planning. It also covers the method of debugging the FOX code and using it in the Front End (WAD).

Author: Anupama Rani Mahto
Company: Infosys Technologies Ltd.
Created on: August 16, 2010

Author Bio
Anupama is working as a Senior Systems Engineer with Infosys Technologies Ltd. He has a good experience in the SAP BI technology and have been involved in the implementation and execution of BW/BI Projects.
Table of Contents

Introduction: ........................................................................................................................................... 3
How to start with Fox Formula: ................................................................................................................ 3
Planning Function Types: .......................................................................................................................... 4
  Standard planning function: ...................................................................................................................... 4
  Formula extension (FOX): ....................................................................................................................... 4
  Custom planning function: ...................................................................................................................... 4
Different program elements of FOX and there use: .................................................................................. 5
Step by step procedure for creation of planning function using Formula extension (FOX): .................. 6
  Step 1 - InfoProvider Creation: ............................................................................................................... 6
  Step 2 – Aggregation Level Creation: .................................................................................................. 7
  Step 3 – Filter Creation: ......................................................................................................................... 8
  Step 4 – Creation of Planning Function: ................................................................................................. 9
  Step 5 – FOX code: ............................................................................................................................... 9
  Step 6 – Planning Sequence Creation: ................................................................................................... 14
Debugging the FOX (Formula extension): ............................................................................................ 15
  Step 1 – Putting break point: ................................................................................................................ 15
  Step 2 – Execution of Program: ........................................................................................................... 15
  Step 3 – Step by step debugging: ......................................................................................................... 16
Front end development to use the FOX formula in reports: ................................................................ 17
  Step 1 – Creation of query: .................................................................................................................. 17
  Step 2 – Web Template creation: ....................................................................................................... 18
  Step 3 – Addition of Button Group: .................................................................................................... 18
Conclusion: .............................................................................................................................................. 21
Related Content ..................................................................................................................................... 22
Copyright .................................................................................................................................................. Error! Bookmark not defined.
Introduction:

BI Integrated Planning provides business experts with an infrastructure for realizing and operating planning scenarios or other applications. Planning covers a wide range of topics from simple data entry to complex planning scenarios.

Business planning allows developers to create and operate planning scenarios or other applications. Business planning provides the following planning tools for these purposes:

- BI Integrated Planning, a solution that is completely integrated into the BI system
- BW-BPS (Business Planning and Simulation)

SAP Business Intelligence Integrated Planning (SAP BI-IP) provides the standard allocation formulas which are used for basic allocation requirement. For more complex business cases, standard formulas are not able to fulfill the requirement and in this case FOX which comes with SAP BI-IP can fulfill the complex development requirements.

How to start with Fox Formula:
• Create Real time cube or partially writable MultiProvider
• Create Aggregation level on the Cube or MultiProvider
• Create a planning enabled query on the aggregation level.
• Also create a Filter (optional) and planning Function on the aggregation level.
• Create a planning sequence on the aggregation level using the filter and planning function created on the same aggregation level.

Planning Function Types:
The planning Functions are functions that support automatic processing of the plan data (such as copying, distributing etc). This processing can happen on existing data or on data entered by user using input ready query. This processing takes place as simulations or background process.

The planning function is of mainly three types:
• Standard planning Functions
• Formula extension (FOX)
• Custom defined planning function

Standard planning function:
The standard planning functions are the predefined planning functions provided by sap to change data. They are easy to use and simplest of the above mentioned types. Standard planning function includes copying, currency translations, deletions, reposting, revaluating, unit conversion etc,

Formula extension (FOX):
FOX is used when the required functionality cannot be achieved using the standard planning function. The FOX provides us a more powerful and flexible approach in creating the planning function. It is similar to programming language with functionalities like loop’s if’s etc.

In the backend an ABAP code is generated for our FOX code written in the planning Function. This generated program gets executed when our planning function is executed. It can be used to cover a wider range of business requirement but is comparatively complex to standard planning function so it should be used only when the required functionality cannot be achieved by standard planning function types. The logic for FOX is defined in the planning function itself.

Custom planning function:
ABAP knowledge is required for creating custom planning function. The ABAP code for this is complex but it is most flexible also compared to the above. It requires more processing time as well. So custom planning Function should be used only when the required functionality cannot be achieved by standard planning function or by FOX.

When Formula extension (FOX) can be used:
• For enhanced mathematical functions for calculating the plan data. It can also be used for comparative and Boolean operations.
• For using some variable input value for complex functionality
• For using the attribute values of some characteristic present in the aggregation level.
• For some kind of changes required with a character string.
• We can also use it for some finance functions.
Different program elements of FOX and their use:

**Statements:** This element comprises of mainly the common elements used in programming like declaration element, loop elements, if else elements etc.

**Variable:** This contains elements to get the variable value or variable related details. Examples for this are VARV (Gets the 1st value entered for variable entry), VARC( Gets the number of values of a BI variable) etc.

**Characteristics and attributes:** This element has commands which are used to get the characteristic and its attributes value (both time dependent and time independent). Some of the elements available here are OBJV (Gets value of variables, inputted at runtime), ATRV(Gets the master data attribute values), TMVL(adds the offset to time characteristic and returns the result) etc

**Character String:** This element contains commands to perform various functionality on strings, like concatenating two strings the element available for this is CONCAT, getting a substring, the element available for that is SUBSTR etc.

**Operators:** It contains the various operators which can be used in FOX, mathematical operator like +, - etc. Comparative operator like =, <> etc and Boolean like is initial etc.

**Mathematical Functions:** The Mathematical Function contains commands to get complex mathematical functionalities like truncating 0’s, sin, cos, tan etc.

**Finance functions:** This element contains commands for some specific finance functions like discount (DISC), pension(PERP) etc.

**Others:** This contains some other commands like finding the maximum or minimum etc. Example: MAX (To find maximum of two values), MIN (to find minimum of two values) etc.
Step by step procedure for creation of planning function using Formula extension (FOX):

Step 1 - InfoProvider Creation: The first step is the creation of Real Time InfoCube. We can also use a partially writable MultiProvider depending on the requirement. The partially writable MultiProvider is created by building the MultiProvider on at least one Real time cube and other InfoProviders. We go for MultiProvider in scenarios where we want to do reporting on more than one InfoProvider say on actuals and on planning. MultiProvider reduces the frequency of errors and granularity of the planned and reported data is the same.

The InfoProvider created can be viewed in the planning Modeler. For this we need to open the Modeler which can be accessed through the transaction code RSPLAN. Once we are in the transaction clicking on the start Modeler takes us to the planning modeler. The modeler opens up in a web page. Once the modeler opens we can view our required InfoProvider by giving the technical name of the InfoProvider.
Step 2 – Aggregation Level Creation: The next step is creation of Aggregation Level. The aggregation level is a subset of the InfoProvider containing the required characteristic and KeyFigures. The data in aggregation level gets aggregated for the characteristic which are not selected in the aggregation Level.
Step 3 – Filter Creation: The next step is creation of filter on the aggregation level. The filter is created by providing a technical name and description for the filter and then selecting the aggregation level on which it should be created. Then we can select one or more characteristics (as required) on which we want to restrict the data as filter characteristics. The filter helps us to restrict the data based on variable entry (at runtime) or direct restriction on characteristic values. The filter can be a restriction on a single value or multiple values.
Step 4 – Creation of Planning Function: Once the filter is created we go on to create the Planning Function on our aggregation level. For writing a FOX code we select the planning function type Formula, provide a technical name and description for the planning function and select the required aggregation level. This Formula planning function type enables us to write FOX(Formula Extension) code.

Step 5 – FOX code: The next step is writing the FOX code. Before writing the code we need to select the characteristic from ‘To Character usage’ for the characteristic which logically belong together and the characteristic values for which we want the KeyFigure changed. This helps in the Block formation. Technically, every planning function that is executed is called internally more than once. For this purpose, the data to be processed selected by filter is divided into blocks. The planning function is called separately for each block.

If this selection is not done the block levels increases, thus the records we require for our functionality might fall in different blocks.
The FOX code is as follows. There are five different type of syntax covered in this code:

1. Data declaration (local Variables) – The data declaration is done as standard type like integer, string or it can be declared as type of the available characteristic.

2. Information derivation (Variables) – The variable function (VARV) helps us to get the 1st value of the variable (global) entered in the Variable screen.

3. Calculations (Operators) – The simple addition command helps us in getting the sum of two months of data.

In case of addition we are using parameters within brackets. This is used to specify characteristics for which we want to change the KeyFigures. In the example shown below we want to add the values of a particular ‘Customer account group’ SBUC for two months 200906 and 200907 for all values of ‘Customer Code’. These parameters contain the characteristics selected in the ‘To Character Usage’
tab, in the same order it is selected there. The parameters within brackets have to be used every time some functionality is performed on any KeyFigure values.

4. Conditional statement – The conditional statement if-else is used to execute a block of statement only if some condition is true. In the example below two months of data is summed, based whichever is greater some assignments are done.

5. Characteristic and attribute – The ATRV command helps us to derive the attribute value for characteristic IC_CNTYCD.

```
DATA Z_ADD TYPE I.
DATA Z_ADD1 TYPE I.
DATA Z_CUSCODE TYPE IC_CUSCOD.
DATA Z_CALMONTH TYPE 0CALMONTH.
DATA Z_CUNTRYCD TYPE STRING.

*GETTING A VARIABLE ENTRY VALUE IN A VARIABLE
Z_CUSCODE = VARV(Z_CUSTOMER).

*ADDING TWO VALUES
Z_ADD = {0DEB_CRE_LC, SBUC, 200905,Z_CUSCODE} + {0DEB_CRE_LC, SBUC, 200906,Z_CUSCODE}.
Z_ADD1 = {0DEB_CRE_LC, SBUC, 200906,Z_CUSCODE} + {0DEB_CRE_LC, SBUC, 200907,Z_CUSCODE}.
{IA_TDSO, SBUC, 200905,0000000003} = Z_ADD.

*FOR SHOWING FUNCTIONALITY OF IF STATEMENT
IF Z_ADD > Z_ADD1.
{0DEB_CRE_LC, SBUC, 200905,0000000003} = Z_ADD.
ELSEIF Z_ADD = Z_ADD1.
{0DEB_CRE_LC, SBUC, 200905,0000000003} = Z_ADD1.
ELSE.
{0DEB_CRE_LC, SBUC, 200905,0000000003} = Z_ADD.
ENDIF.

*USE ATTRIBUTES
Z_CUNTRYCD = ATRV(IC_CNTYCD,Z_CUSCODE).

IF Z_CUNTRYCD = 'IN'.
{0DEB_CRE_LC, SBUC, 200905,0000000003} = Z_ADD - 100.
ENDIF.
```

The second FOX code is as follows. The syntax covered here are:
1. **Loop** – The loop statement helps us to loop at all the data based on some local variable. In the example shown we loop through all the data based on Calendar month. The other type of loop possible in FOX is Do loop.

2. **Mathematical function** – There are many mathematical function possible in FOX like Sine, Truncate etc. The ‘ROUND’ command helps us to round off a value to the required no of places. Here we are rounding off to ‘0’ decimal places.

3. **Calling Function Module** – We can call function module in FOX code also. For this we need to create a Function Module in the transaction code SE37 with the Import and Export parameters we require. Then we go to transaction SM31 and make an entry for the Function Module in the table RSPLF_FDIR. Once this is done the Function Module is available to be called in the FOX code. The import and Export parameters should be in sync with the DataType supported by FOX. In the example shown below we call ZIP_TEST function module exporting Z_CUSCODE value and import Z_IC_GEOSEG value. The imported value is used further in the code.

4. **Finance Functions** – The finance function covers some commonly used finance functions like currency translation, discount calculation etc. Here is an example how to calculate the discount of 10% for 2 years.

```plaintext
DATA Z_ADD TYPE F.
DATA Z_CUSCODE TYPE IC_CUSCOD.
DATA Z_CALMONTH TYPE 0CALMONTH.
DATA Z_IC_GEOSEG TYPE IC_GEOSEG.
DATA Z_ROUND TYPE I.
DATA Z_ROUND1 TYPE F.
DATA Z_DISC TYPE F.

*TESTING FOR EACH FUNCTIONALITY TO UPDATE KEYFIGURE VALUE IN AN ADDITIVE WAY
FOREACH Z_CALMONTH.
  Z_ADD = Z_ADD + {0DEB_CRE_LC, SBUC, Z_CALMONTH,Z_CUSCODE}.
{0DEB_CRE_LC, SBUC, Z_CALMONTH,Z_CUSCODE} = Z_ADD.

*FOR GETTING ROUNDED OFF VALUES
Z_ROUND1 = ROUND(Z_ADD,0).
{IA_TDSO, SBUC, Z_CALMONTH,Z_CUSCODE} = Z_ROUND1.

*FOR CALLING A FUNCTION MODULE TO GET SOME LOOKUP VALUE
CALL FUNCTION ZIP_TEST
  EXPORTING  Z_CUSCODE = Z_CUSCODE
  IMPORTING   Z_IC_GEOSEG = Z_IC_GEOSEG.

  IF Z_IC_GEOSEG <> 'INDIA'.
    {0DEB_CRE_LC, SBUC, Z_CALMONTH,Z_CUSCODE} = Z_ADD - 200.
  ENDIF.

*FOR FINENCIAL FUNCTIONS
```

Looping at the records based on calmonth

For getting value rounded off to ‘0’ decimal places

Calling Function Module ZIP_TEST
ENDFOR.

\[ Z_{\text{DISC}} = \text{DISC}([\text{0DEB_CRE_LC,SBUC,200909,#}], 10, 2). \]

\[ \{\text{IA_TDSO,SBUC,200909,#}\} = Z_{\text{DISC}} \]

End of Loop

Calculating the discount value at 10% for 2 years

‘#’ can be used whenever a characteristics value is unassigned or blank.
**Step 6 – Planning Sequence Creation:** The next step is creation of planning sequence. Planning function is used to bundle different planning functions (from the same aggregation level or different aggregation level) together. For creation of Planning Sequence we need to give a Technical name and description for the planning function. Then we need to select ‘Add Step to planning Function’ if we want to execute some planning Function or we select ‘Add step to Input Template’ if we just want to view data in an aggregation level based on some filter. In case we select ‘Add Step to planning Function’ we need to select the aggregation level, Filter and Planning Function. Else if we select ‘Add step to Input Template’ we just need to select the Aggregation level and the filter.

The input template of the planning sequence gives the initial view of the data based on the filter while the ‘Planning function’ step on execution modifies the data based on the functionalities performed by the selected Planning Function. The ‘Planning function’ step can be executed by pressing ‘Execute’ or ‘Execution with trace’. The ‘Execute’ button executes the planning sequence or the selected step and modifies the data which can be saved by pressing the ‘Save’ button. In case we press ‘Execution with trace’, we get detailed information on which all data records were changed. And thus we can do a detailed analysis of the Planning Function.

The variable input value needs to be provided before execution of planning sequence if the variable is a mandatory variable or its value is used in the planning function.
Debugging the FOX (Formula extension):

**Step 1 – Putting break point:** The FOX code can be debugged in case of any discrepancies, when the output is not coming as expected. The debugging can be done by putting a break-point in the FOX formula.

![Debugging Step 1](image1)

**Step 2 – Execution of Program:** Then we go to the transaction code SE38 and execute the program RSPLS_PLSEQ_EXECUTE, providing the planning sequence name there. We can save the details as variants for further use. We can also save the data after the planning sequence execution. This can be done by checking the checkbox shown below. If the checkbox is checked once the planning sequence code is executed the changed data gets saved in the planning InfoProvider.

![Execution Step 2](image2)
Step 3 – Step by step debugging: Now the FOX code can be debugged. And the exact step by step functionality can be checked for.
**Front end development to use the FOX formula in reports:**

The planning function should be used in reports for the end user to plan and process the data using the planning function.

The steps for front end development are:

**Step 1 – Creation of query:** An Input ready query needs to be created on the aggregation level. The query designer is used to display the data and also provide some additional calculations and restrictions on the KeyFigures. The query is made input ready by checking the radio button Input-Ready (Relevant for locking) in the properties if the KeyFigure which have to be Input Relevant.
Step 2 – Web Template creation: Next the Web Template is created with the above created Input Ready query. It is not necessary to set the data provider initially; it can be set dynamically at runtime also. The analysis web item of the Web template enables us to view the query data and manipulate the data according to our requirement.

Step 3 – Addition of Button Group: A button group is also inserted in the web item to implement the planning function and to save data. We can enable our planning function with other Web Items as well.

The settings for the button group are as follows:
The command EXEC_PLANNING_SEQ is selected and the required planning sequence is selected in it.

To save data

WAD output with the button group and the analysis item after the Planning Sequence is executed:
<table>
<thead>
<tr>
<th>Customer Code</th>
<th>Geographic Segment</th>
<th>Cat. year / month</th>
<th>Fiscal Year</th>
<th>Calendar year</th>
<th>Time</th>
<th>User Name</th>
<th>LC-Amount</th>
<th>Target EOS for IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>India</td>
<td>01.2009</td>
<td>2009</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>$57,409,007.87</td>
<td>57,409,208</td>
</tr>
<tr>
<td></td>
<td></td>
<td>02.2009</td>
<td>2009</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>$114,369,878.40</td>
<td>114,369,078</td>
</tr>
<tr>
<td></td>
<td></td>
<td>03.2009</td>
<td>2009</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>$164,695,599.70</td>
<td>164,695,799</td>
</tr>
<tr>
<td></td>
<td></td>
<td>04.2009</td>
<td>2010</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>$46,576,599.00</td>
<td>46,576,786</td>
</tr>
<tr>
<td></td>
<td></td>
<td>05.2009</td>
<td>2010</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>$95,067,599.00</td>
<td>95,067,786</td>
</tr>
<tr>
<td></td>
<td></td>
<td>06.2009</td>
<td>2010</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>$148,736,599.00</td>
<td>148,736,786</td>
</tr>
<tr>
<td></td>
<td></td>
<td>07.2009</td>
<td>2010</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>$193,267,750.00</td>
<td>193,267,950</td>
</tr>
<tr>
<td></td>
<td></td>
<td>08.2009</td>
<td>2010</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>09.2009</td>
<td>2010</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10.2009</td>
<td>2010</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>$238,875,834.00</td>
<td>238,876,034</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11.2009</td>
<td>2010</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>$289,463,032.79</td>
<td>289,463,233</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12.2009</td>
<td>2010</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>0080000003</td>
<td>Not assigned</td>
<td>05.2009</td>
<td>2009</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>$0.00</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>06.2009</td>
<td>2009</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>$0.00</td>
<td>0</td>
</tr>
<tr>
<td>0010D1</td>
<td>North America</td>
<td>04.2008</td>
<td>2009</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>$0.00</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>05.2009</td>
<td>2010</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>$0.00</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>06.2009</td>
<td>2010</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>$0.00</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>07.2009</td>
<td>2010</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>$0.00</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>08.2009</td>
<td>2010</td>
<td>2006</td>
<td>#</td>
<td>lot assigned</td>
<td>$0.00</td>
<td>0</td>
</tr>
</tbody>
</table>
Conclusion:
Using FOX (Formula extension) we can handle complex functionalities which cannot be handled by the standard planning function types provided by SAP. The planning data gets modified when the planning sequence containing the FOX planning function is executed. Implementing the FOX formula is complex as compared to the standard planning function types, so it should be used only when the required functionalities cannot be achieved by standard planning function types.
Related Content
www.sdn.sap.com
help.sap.com
http://help.sap.com/saphelp_nw70/helpdata/en/44/21de357641648fe10000000a1553f7/content.htm
For more information, visit the EDW homepage.
Disclaimer and Liability Notice

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

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

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