Customizing Built In Formulas

Applies to:

Software Component: SAP_BW. For more information, visit the EDW homepage.

Summary

This document demonstrates the performance of field routine as compared to formulas used in an update rule/ transformation. Also it provides a simplified mechanism to customize the formula code and a step by step solution to deal with complex scenarios. At the end, there is an extensive list of useful methods along with classes on which the built in formulas are made.

Author: Richa Sahai

Company: Infosys Limited

Created on: 25 August 2011

Author Bio

Richa Sahai is working at Infosys Limited as Systems Engineer since 2009.

Table of Contents

Need for Customizing

1. Better performance of routine over formula
2. Dealing with complex scenarios

Performance of Routine Vs Formula:

There are scenarios in BW where millions of records are to be loaded especially during full repairs and reducing data load time is a priority to increase the performance of a given task. Also, using inbuilt formula seems to be a simpler solution rather than writing an ABAP code in update rule/transformation. However, using a routine gets a job done more quickly than in built formula. The below case study depicts the same.

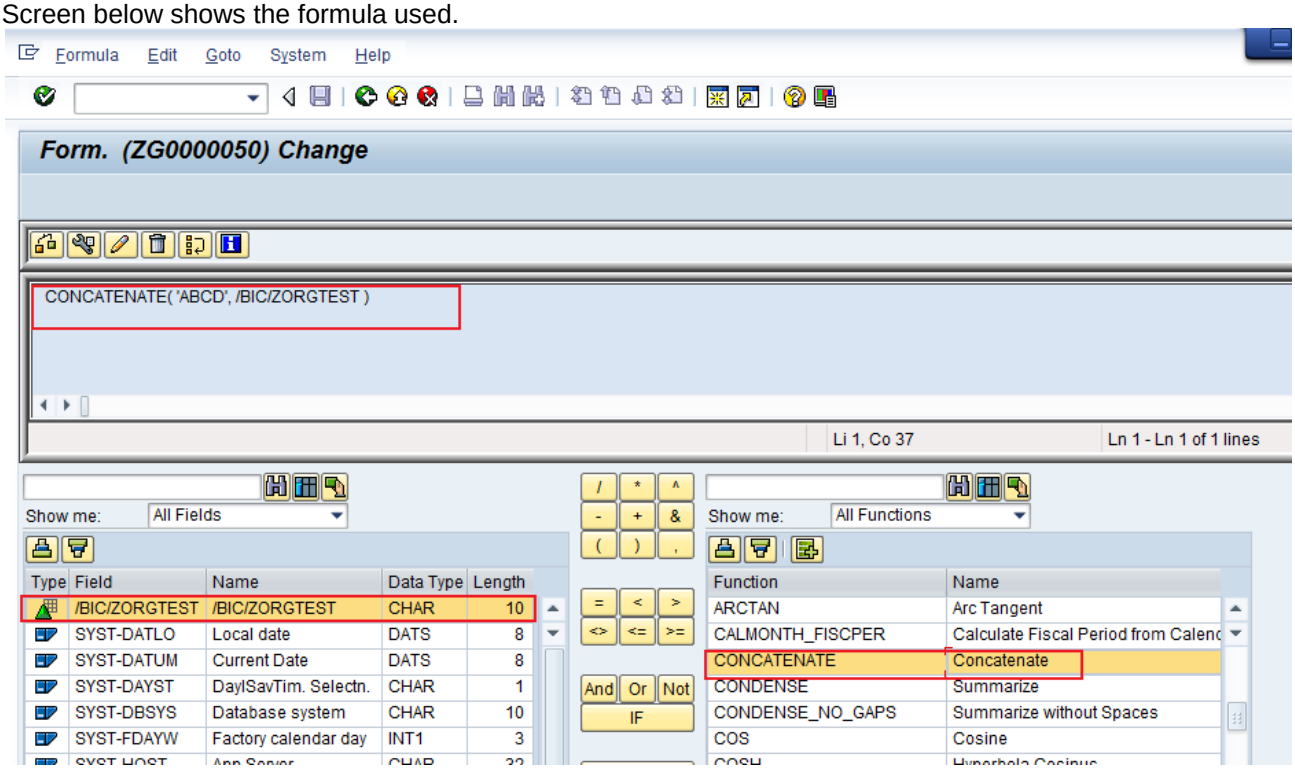
Case Study:

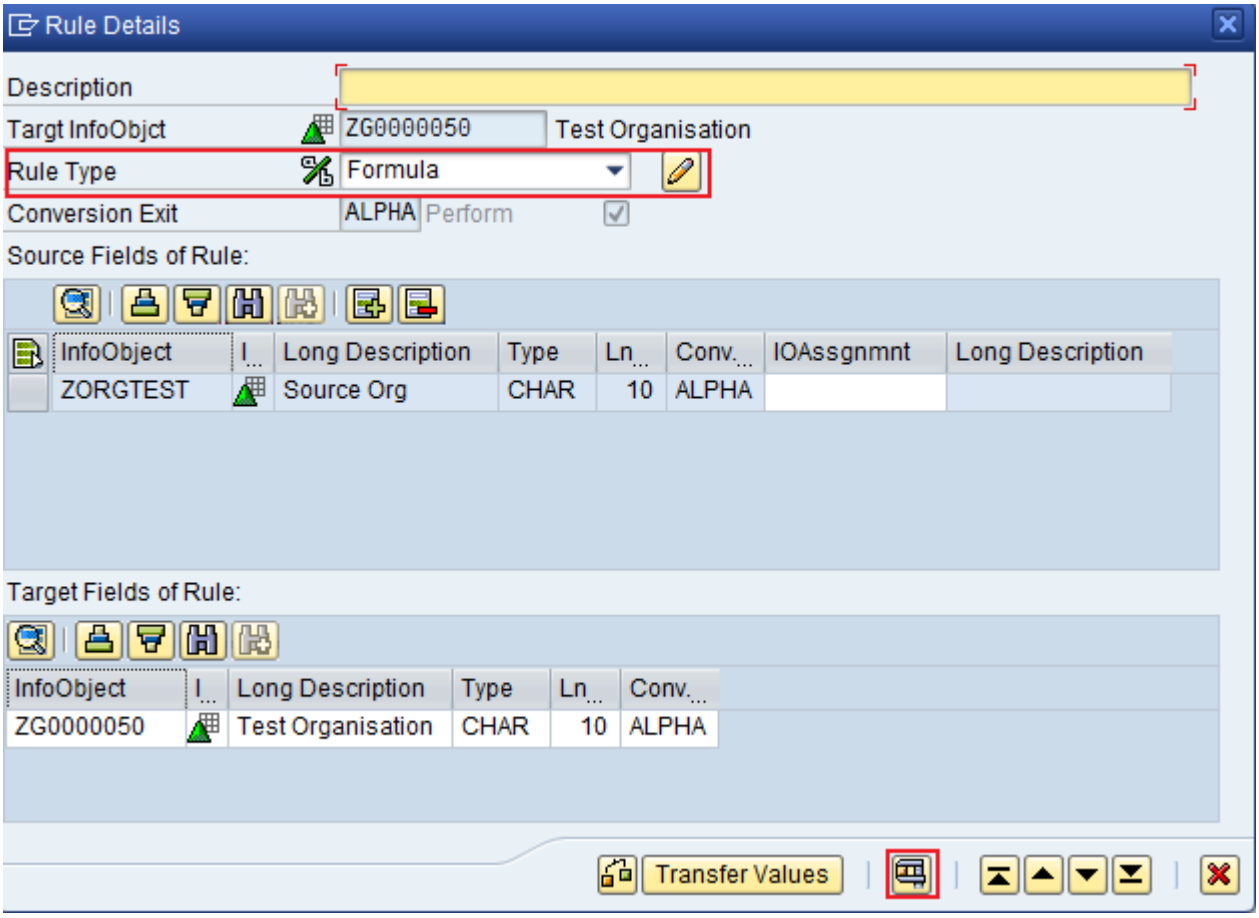
To concatenate a string with a source field.

This could be achieved by two ways:

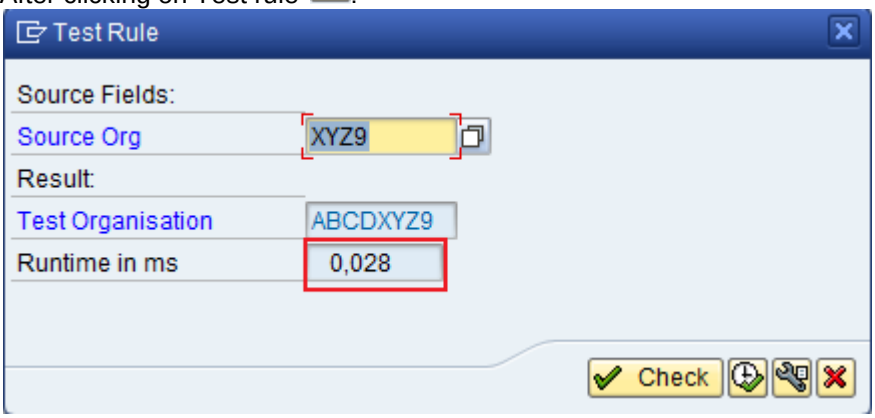
1. **Using a built in Formula :**

   Screen below shows the formula used.





After clicking on Test rule  :

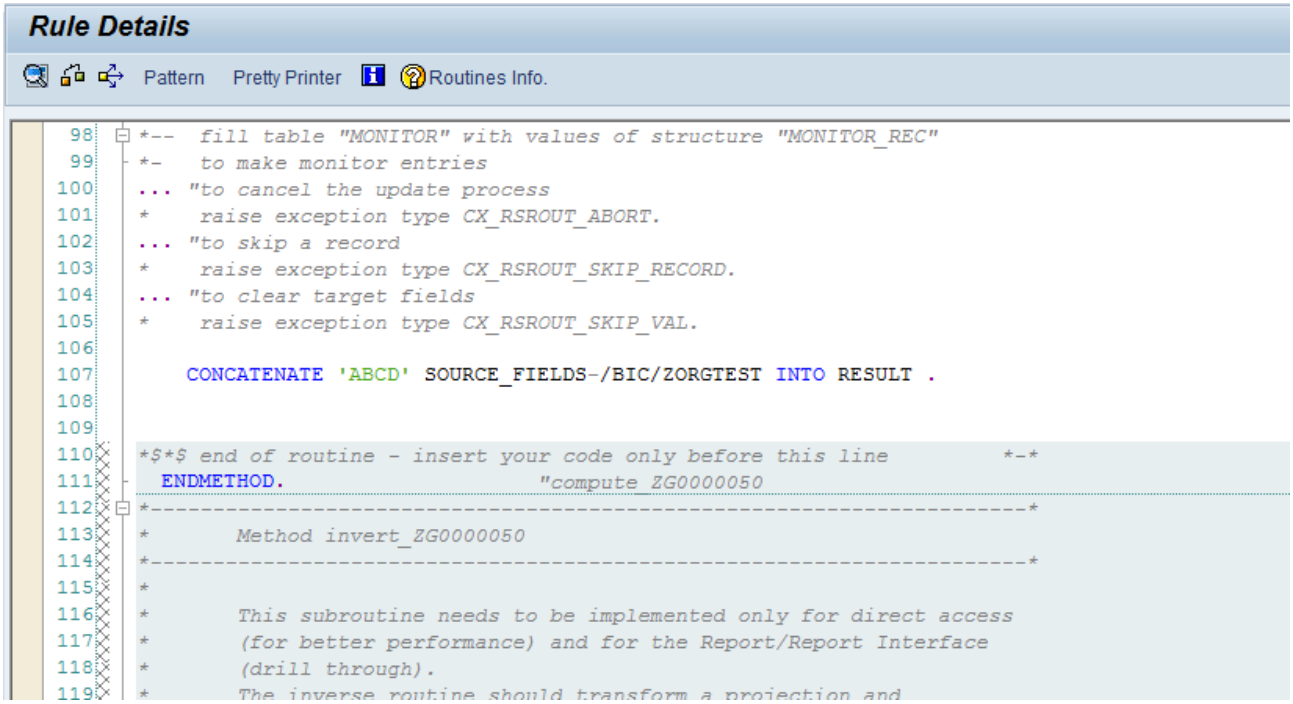


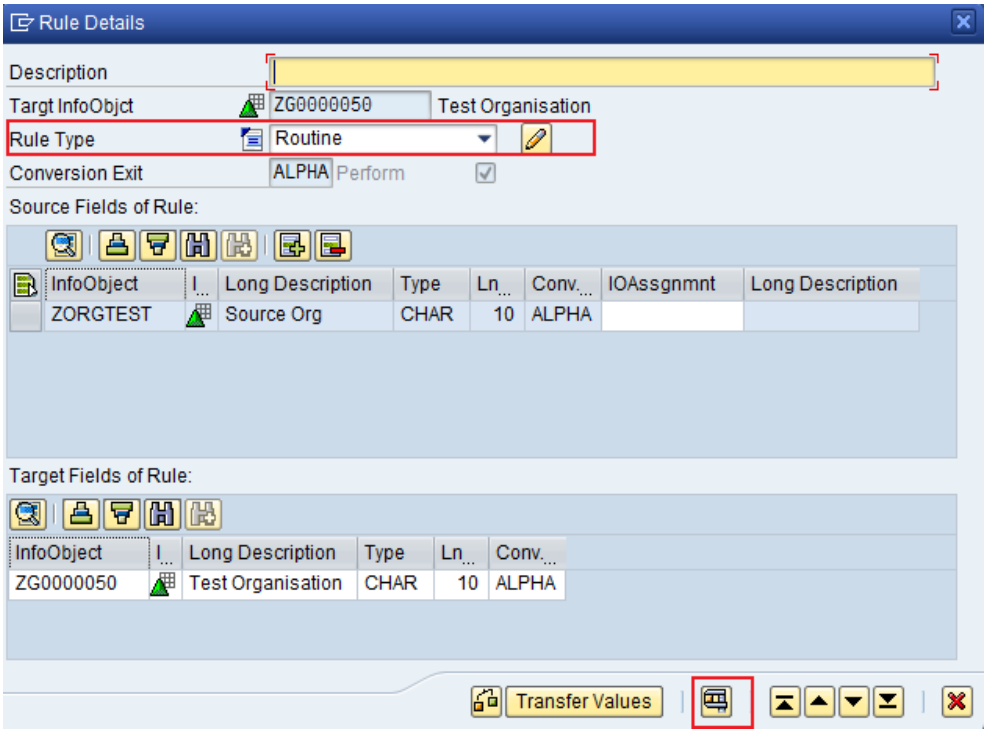
Time taken: 0.028ms

For 50000 records, loading this field will take =1400 ms

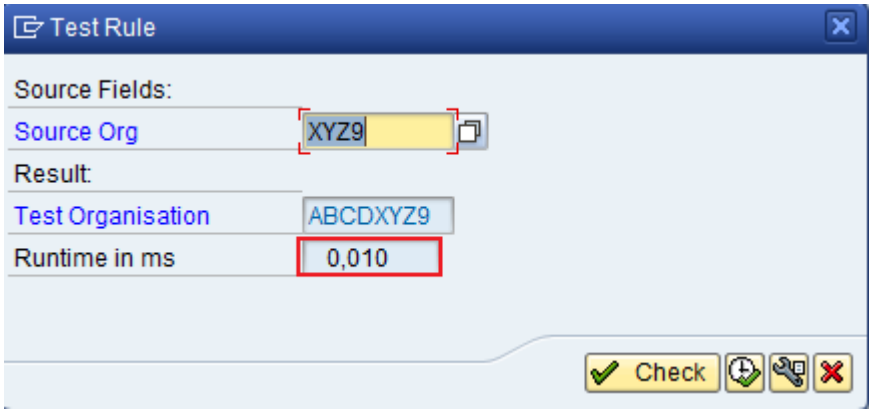
2. Using a Routine

Screen below shows the routine used.





After clicking on Test rule :



Time taken: 0.010ms

For 50000 records, loading this field will take = 500ms seconds

Time Taken By Formula = 1400ms

Time Taken By Routine = 500ms

Time Saved = 900 ms

% Time Saving = Time saved/ Time taken by routine

= 900/500 *100

= 180 %

Thus for a same given task, the performance of a data load could be improved drastically if we replace formula by a routine.

Why this Difference?

1. Formulas are not compiled directly (like ABAP-Routines) and are therefore much slower.

2. ABAP is quicker, and perhaps not only slightly, as we can code in two instructions something that will be coded in ten by the formula, just because the built in formula needs to comply with every possibility while we will put the necessary checks required.

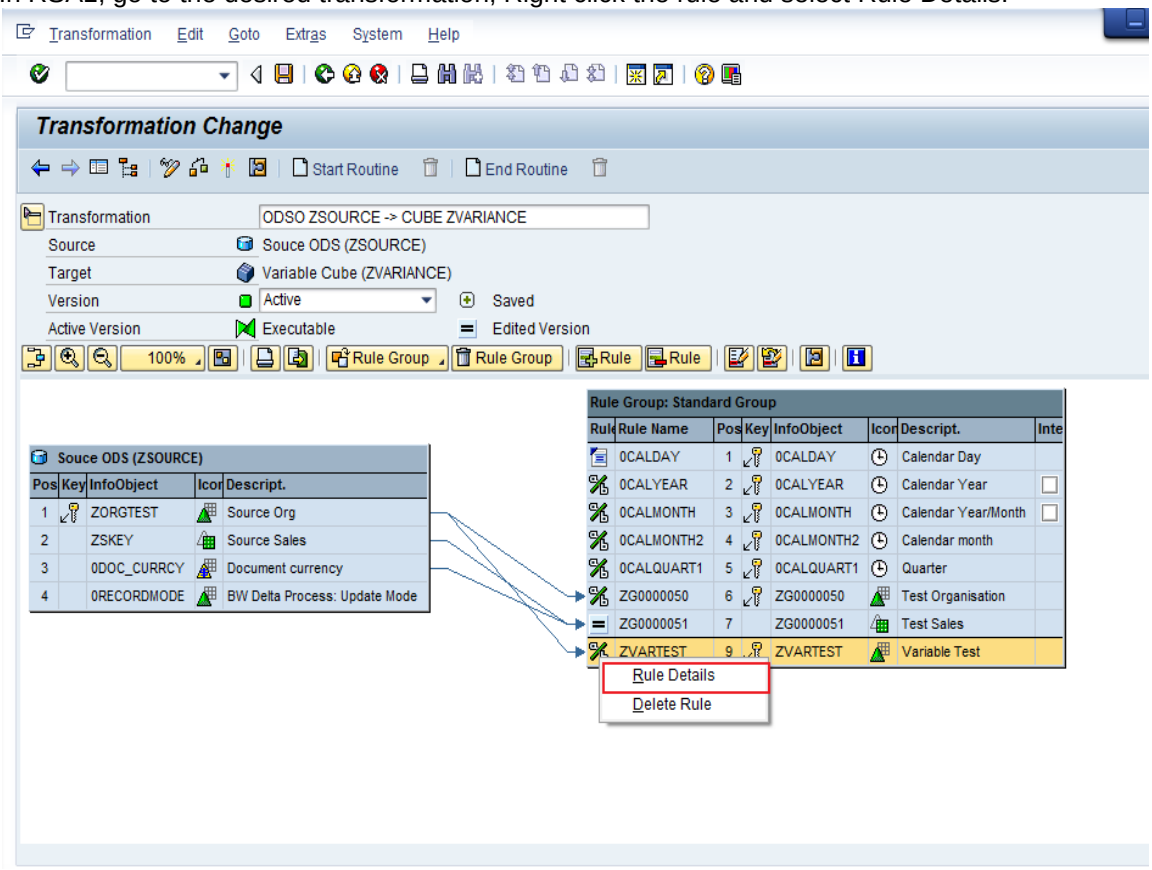
Customizing the Code:

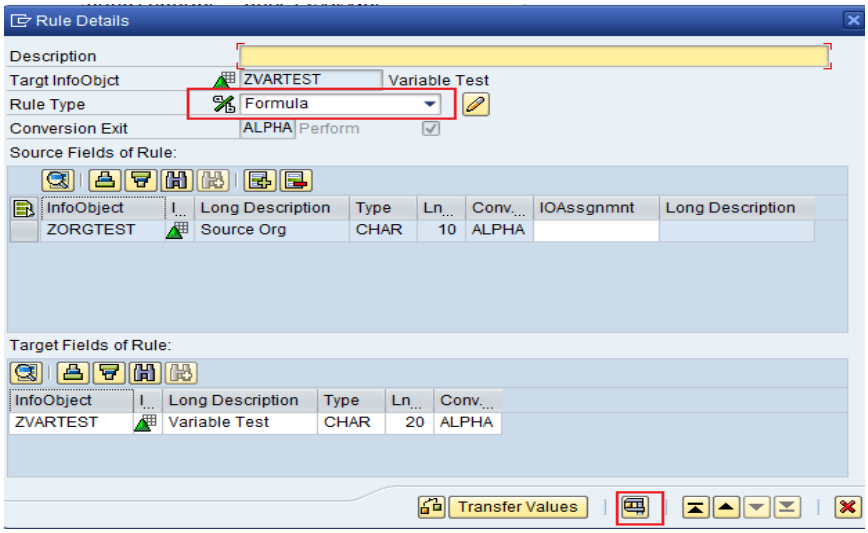
Suppose,

We need to find day corresponding to the current system date.

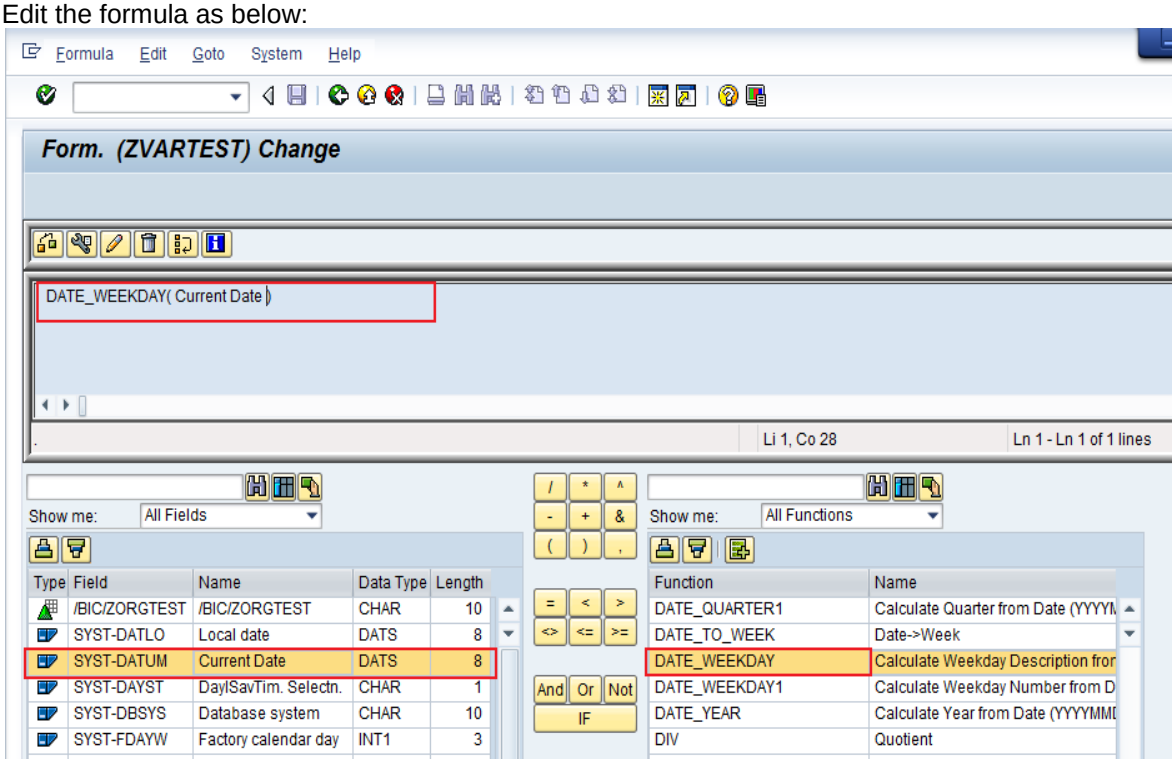
Solution for this by using FORMULA:

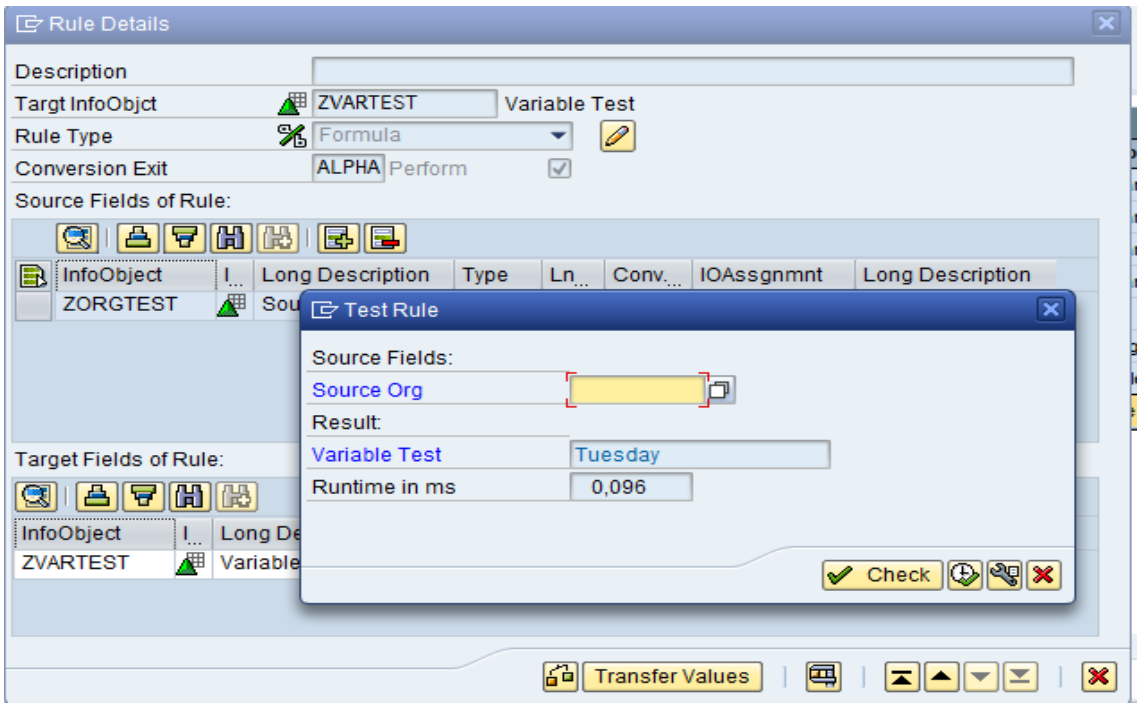
In RSA1, go to the desired transformation, Right click the rule and select Rule Details.





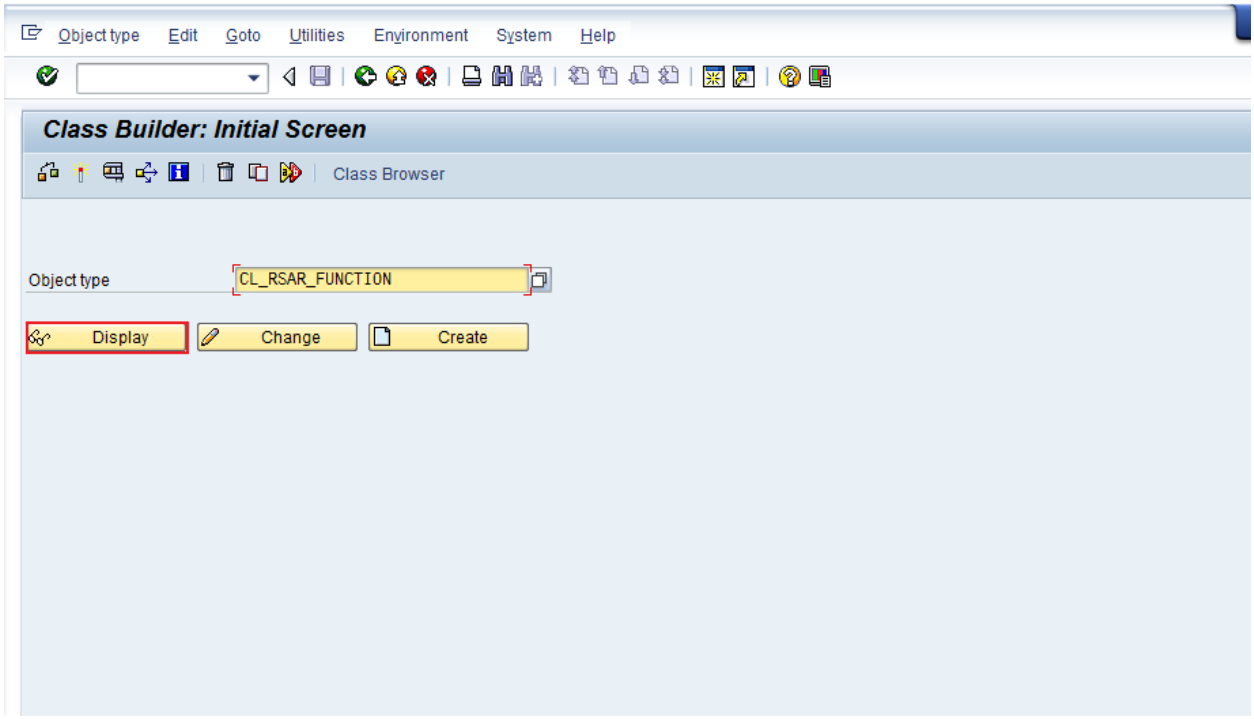
Edit the formula as below:





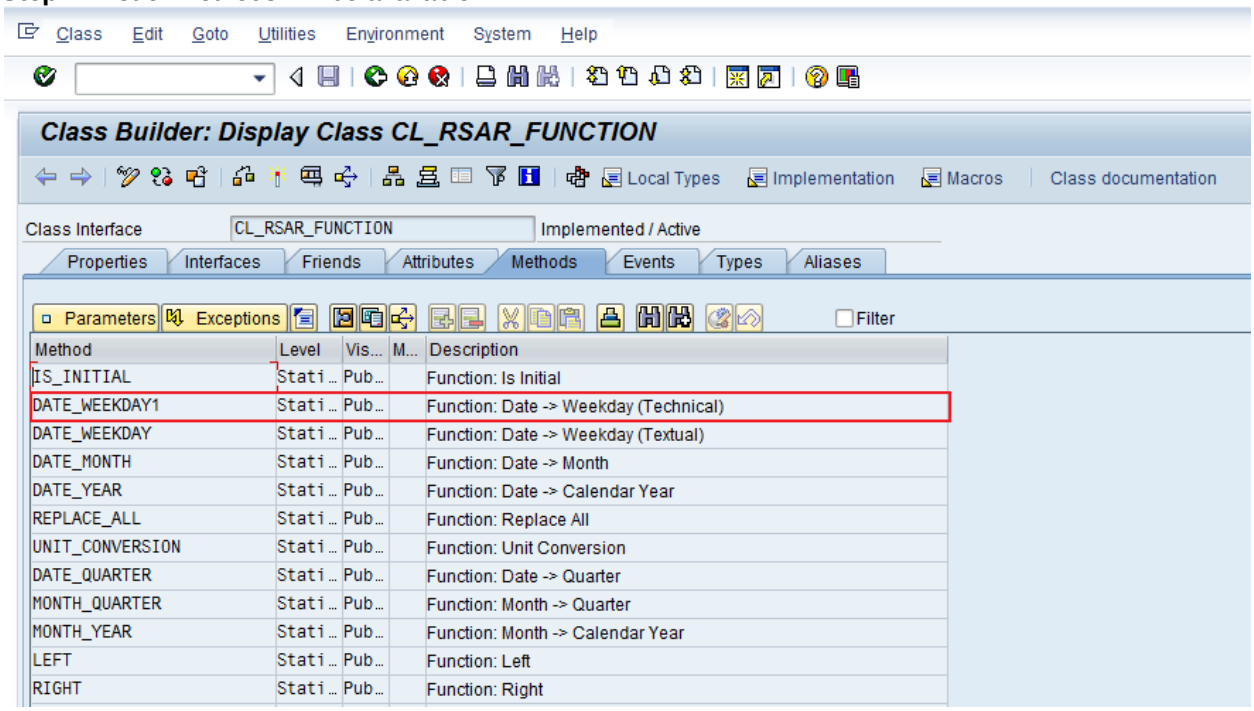
Solution for this by using Routine:

Step 1: Go to SE24 and enter Object Type: CL_RSAR_FUNCTION (Class which contains this function method).



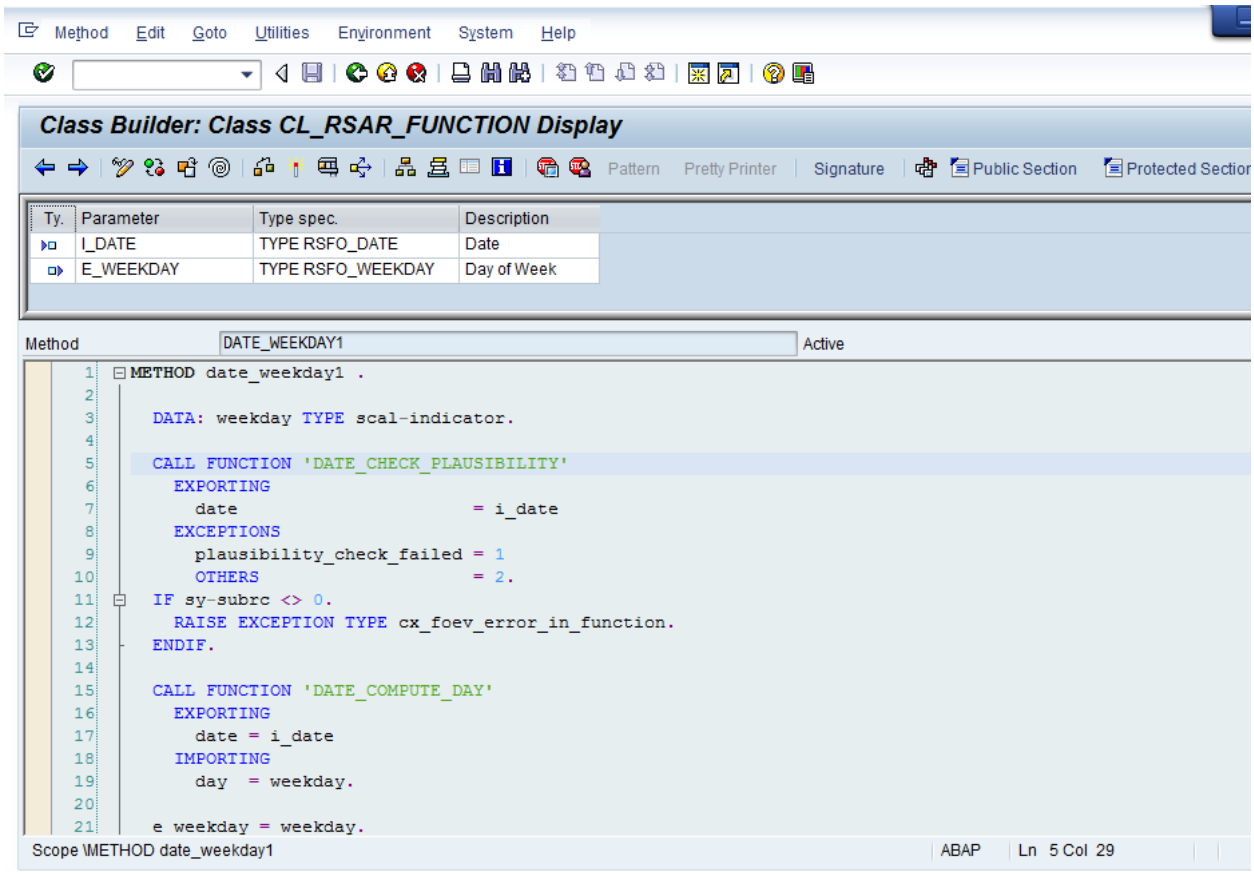
Choose Display.

Step 2: List of methods will be available.



Double click the method name.

Step 3: Examine and copy the code.



Since we are taking current system date, date_check_plausibility seems an unnecessary check. Sometimes we are incorporating such checks in DTP routine while data gets loaded / Start routine. Then incorporating field level checks may become redundant and time consuming.

Step 4: Customizing the code:

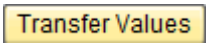
Change the RESULT field.

```
DATA: weekday TYPE scal-indicator.

    CALL FUNCTION 'DATE_COMPUTE_DAY'
    EXPORTING
      date = SOURCE_FIELDS-/BIC/RL0000053
    IMPORTING
      day  = weekday.

  CASE weekday.
    WHEN '1'.
      RESULT = 'MONDAY'.
    WHEN '2'.
      RESULT = 'TUESDAY'.
    WHEN '3'.
      RESULT = 'WEDNESDAY'.
```

```
    WHEN '4'.
      RESULT = 'THURSDAY'.
    WHEN '5'.
      RESULT = 'FRIDAY'.
    WHEN '6'.
      RESULT = 'SATURDAY'.
    WHEN '7'.
      RESULT = 'SUNDAY'.
  ENDCASE.
```

Copy the code to your target field by  edit button. Save the code. Click on  and activate the transformation .

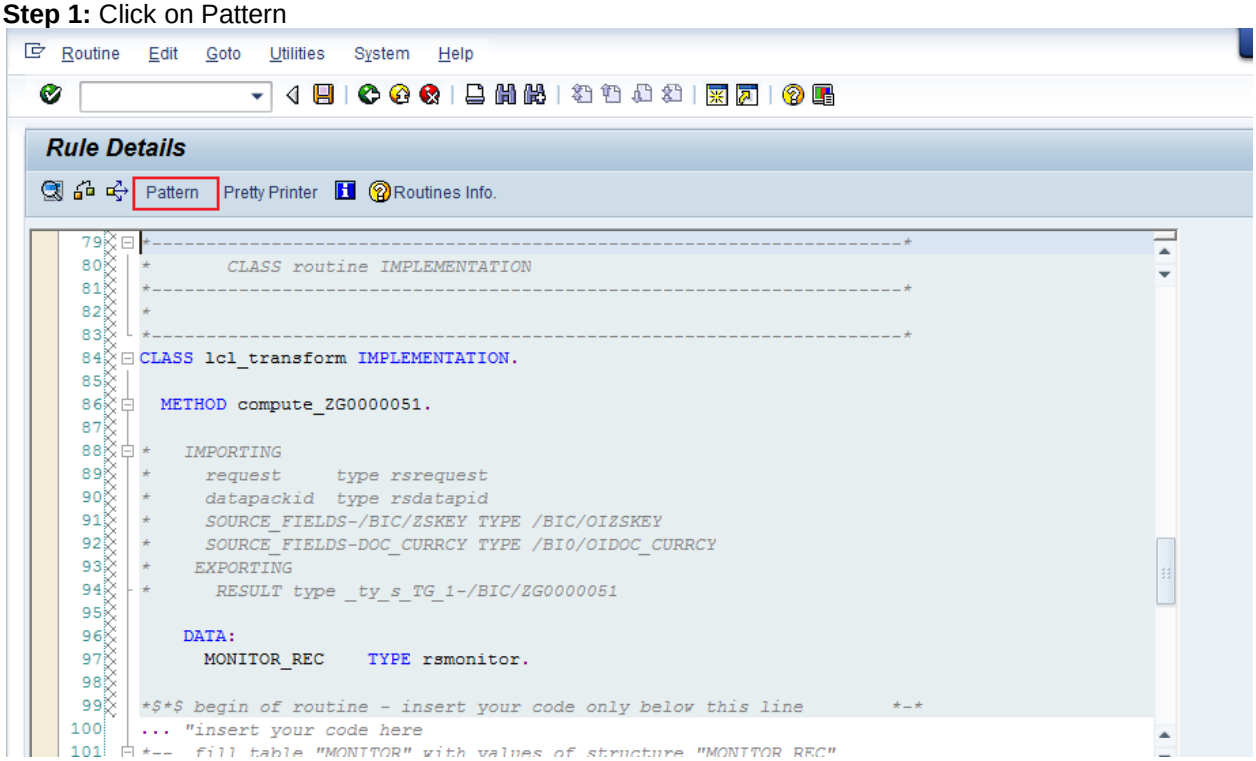
After that, click on Test rule/F9 ,  and execute .Note the difference.

Dealing with Complex Scenarios

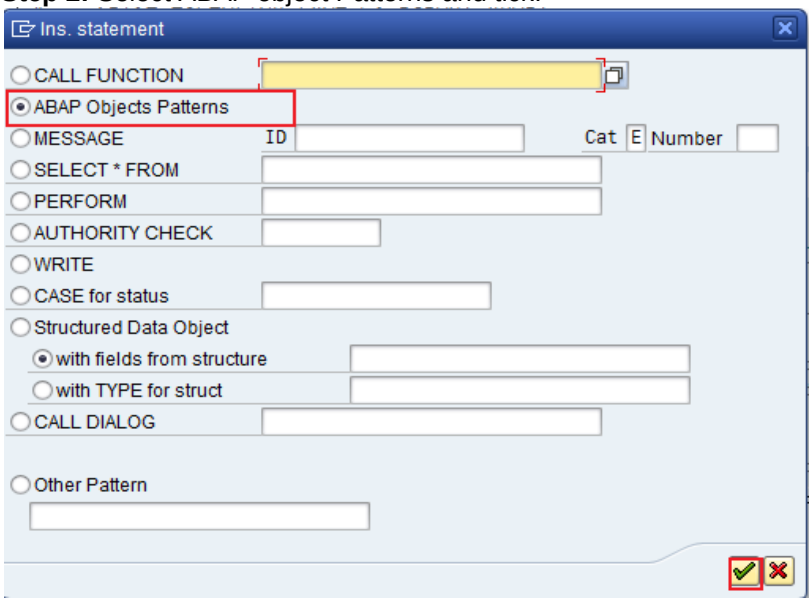
There are other scenarios where the robust code of in built formulas is required (That is we cannot eliminate any checks). Also, simply using this formula does not serve the purpose. The situation could be any like two or three formulas are required one after the other or formula is required within some complex code.

Suppose we need the fraction part of a value after dividing the source field by 2:

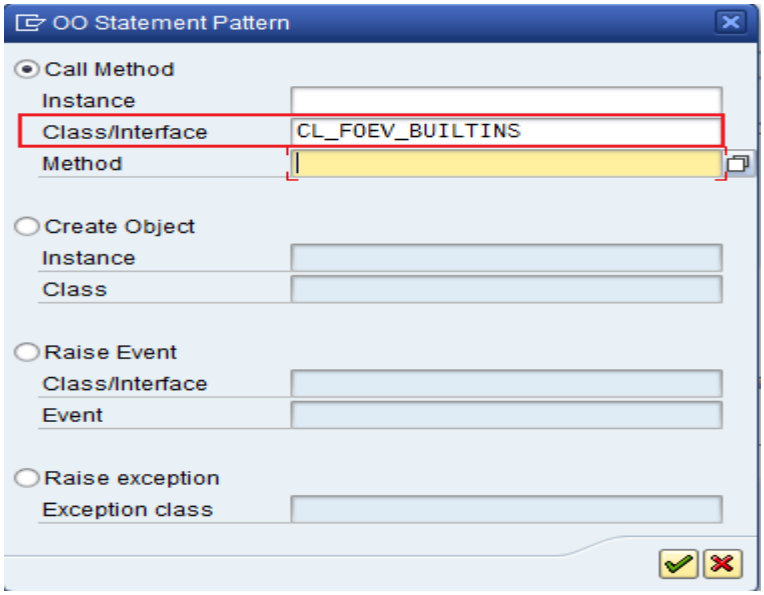
Step 1: Click on Pattern



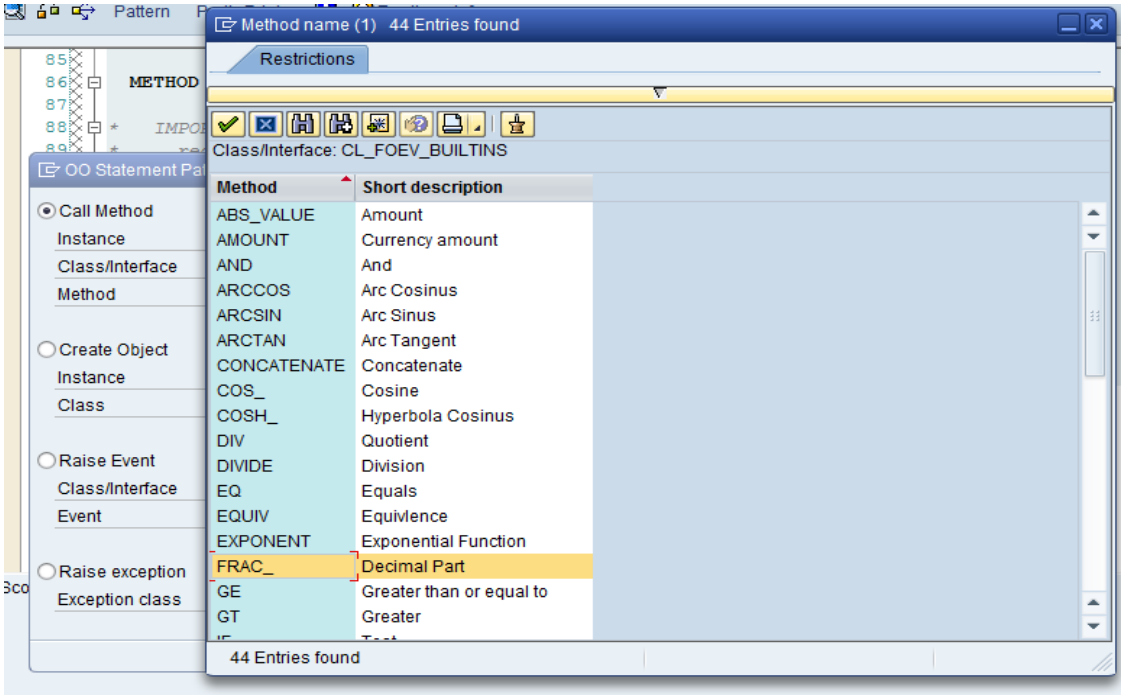
Step 2: Select ABAP object Patterns and tick.



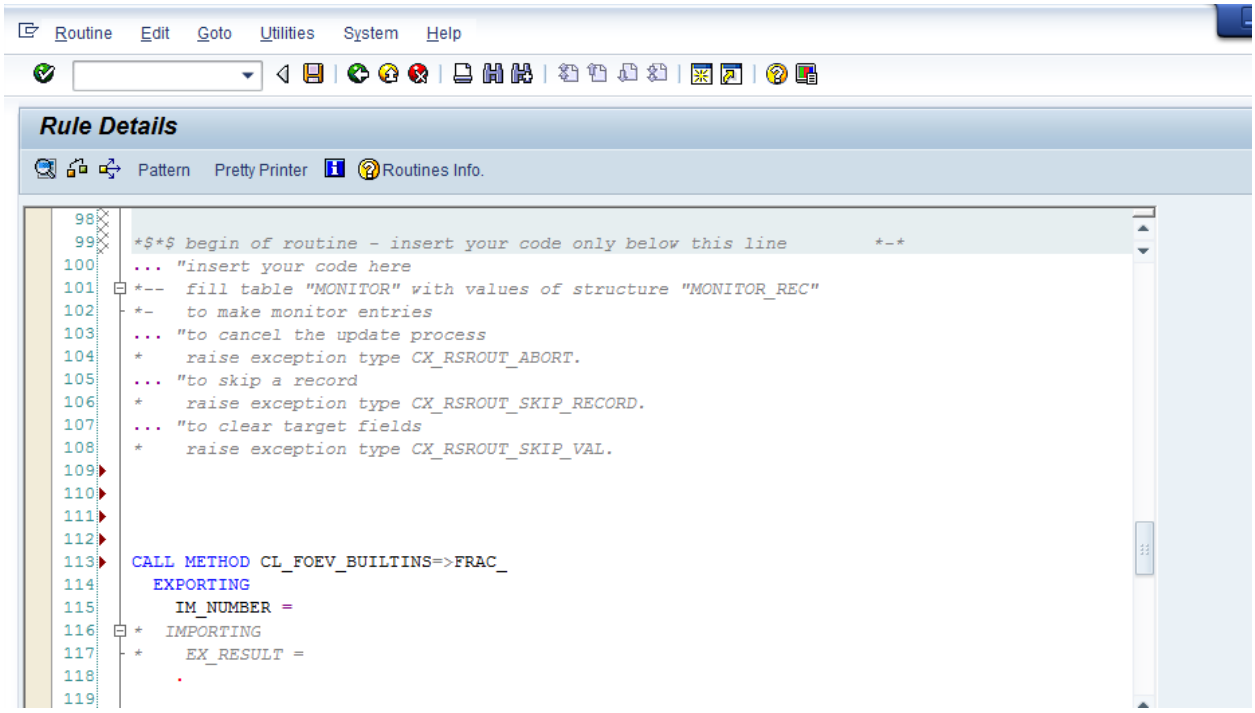
Step 3: Enter Class as CL_FOEV_BUILTINS



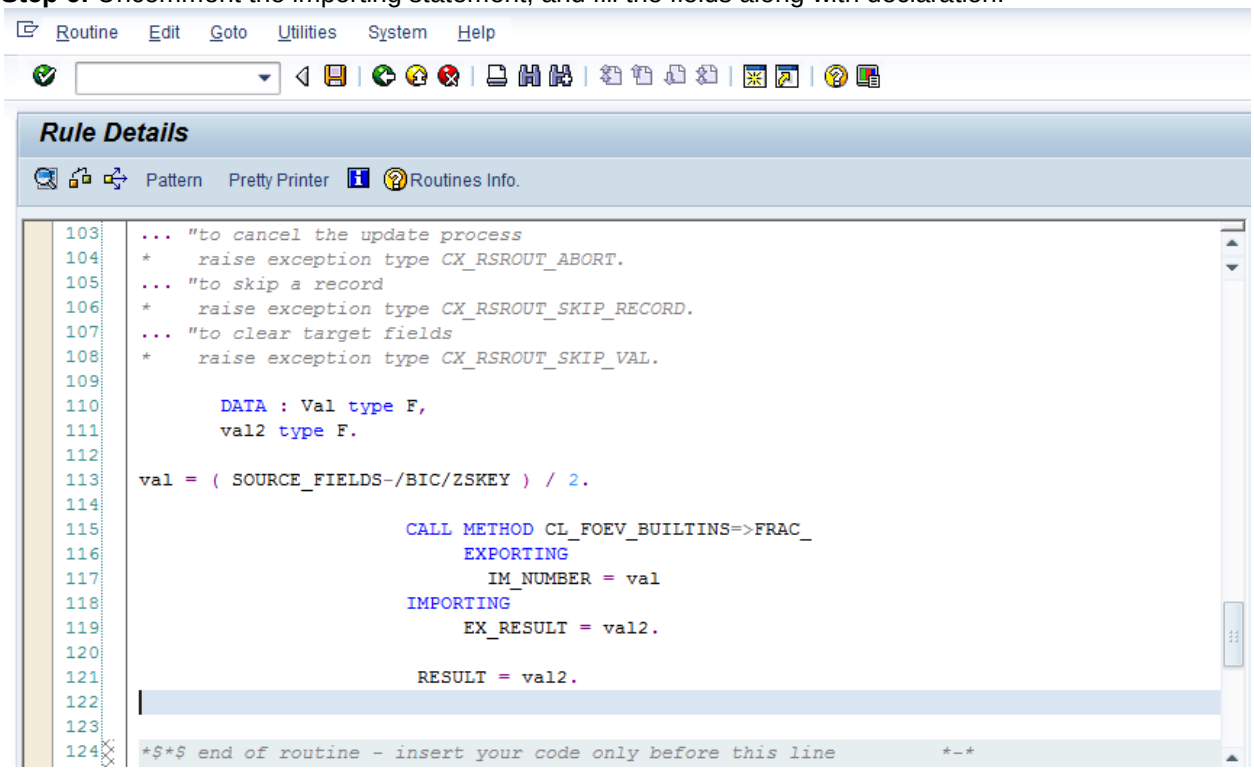
Click on Method and press F4.



Step 5: Choose the desired function, and . Then below screen will appear:



Step 6: Uncomment the importing statement, and fill the fields along with declaration:



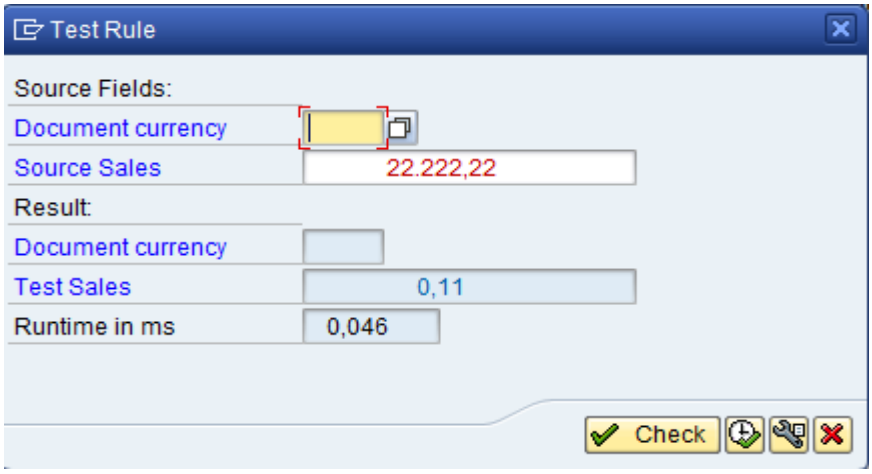
Repeat the same process for the next formula required or modify the code further as per needs and assign it to result.

Save , and Click on Transfer Values and activate the transformation .

Thus,

Like this we can use as many formulas as we want by calling one after the other and include any other calculations we want to in between.

OUTPUT:



List of Useful Methods with Classes:

CL_FOEV_BUILTINS:

No	Method Name	Description
1	ABS	Amount
2	AND	And
3	ARCCOS	Arc Cosinus
4	ARCSIN	Arc Sinus
5	ARCTAN	Arc Tangent
6	CONCATENATE	Concatenate
7	COS	Cosine
8	COSH	Hyperbola Cosinus
9	DIV	Quotient
10	EXP	Exponential Function
11	FRAC	Decimal Part
12	IF	Test
13	LOG	Natural Logarithm
14	LOG10	Logarithm for Basis 10
15	MOD	Remaining
16	NOT	Not
17	OR	Or
18	SIGN	Sign
19	SIN	Sine

20	SINH	Hyperbola Sinus
21	SQRT	Root
22	TAN	Tan
23	TANH	Hyperbola Tan
24	TOUPPER	Uppercase Letters
25	TRUNC	Integer Part
26	POWER	Raise to a power
27	SUBSTRING	Part of Character String
28	TOUPPER	Uppercase Letters
29	SQUARE_ROOT	Root
30	TRUNC_	Integer Part

CL_RSAR_FUNCTION:

No	Method Name	Description
1	ABORT_PACKAGE	Cancel Package
2	ADD_TO_DATE	Add Day to a Date
3	CALMONTH_FISCPER	Calculate Fiscal Period from Calender Month (YYYYMM -Y YYYYPPP)
4	CONDENSE	Summarize
5	CONDENSE_NO_GAPS	Summarize without Spaces
6	DATECONV	Date conversion
7	DATE_DIFF	Date Difference
8	DATE_FISCPER	Calculate Posting Period from Date(Year + Posting Period)
9	DATE_FISCPER3	Calculate Posting Period from Date(Posting period)
10	DATE_FISCYEAR	Calculate Fiscal Year from Date
11	DATE_HALFYEAR	Calculate Half Year from Date (YYYYMMDD-> H)
12	DATE_MONTH	Calculate Month from Date (YYYYMMDD -> YYYYMM)
13	DATE_MONTH2	Calculate Month from Date (YYYYMMDD -> MM)
14	DATE_QUARTER	Calculate Quarter from Date (YYYYMMDD -> YYYYQ)
15	DATE_QUARTER1	Calculate Quarter from Date (YYYYMMDD -> Q)
16	DATE_TO_WEEK	Date->Week
17	DATE_WEEKDAY	Calculate Weekday Description from Date
18	DATE_WEEKDAY1	Calculate Weekday Number from Date
19	DATE_YEAR	Calculate Year from Date (YYYYMMDD -> YYYY)
20	FISCPER_CALMONTH	Calculate Calendar Month from Fiscal Period (PPP -> YYYYMM)
21	FISCPER_FISCYEAR	Calculate Fiscal Year from Booking Period
22	IS_INITIAL	Check for Initial Value
23	LAST_WORKINGDAY_MONTH	Calculate Last Work Day for Month
24	LAST_WORKINGDAY_YEAR	Calculate Last Work Day for Year
25	LEFT	First N Chars
26	L_TRIM	Delete Leading Spaces
27	MONTH2_HALFYEAR	Calculate Half Year from Month (M or MM -> H)
28	MONTH2_QUARTER1	Calculate Quarter from Month (M or MM -> Q)
29	MONTH_HALFYEAR	Calculate Half Year from Month (YYYYMM->H)

30	MONTH_QUARTER	Calculate Quarter from Month (YYYYMM-> YYYYQ)
31	MONTH_QUARTER1	Calculate Quarter from Month (YYYYMM-> Q)
32	MONTH_YEAR	Calculate Year from Month (YYYYMM-> YYYY)
33	NEGATIVE	Reverse +/- Signs
34	QUARTER1_HALFYEAR	Calculate Half Year from Quarter (Q or YYYYQ -> H)
35	QUARTER_HALFYEAR	Calculate Half Year from Quarter (YYYYQ->H)
36	QUARTER_YEAR	Calculate Year from Quarter (YYYYQ-> YYYY)
37	REPLACE_ALL	Replace all
38	REPLACE_FIRST	Replace First
39	R_TRIM	Delete End Spaces
40	SHIFT_LEFT	Move Left
41	SHIFT_RIGHT	Move Right
42	SKIP_RECORD	Skip Record
43	SKIP_RECORD_AS_ERROR	Skip Record (with Error Message to Monitor)
44	STR_LEN	Character String Length
45	WEEK_TO_1ST_DAY	Week->Date
46	WORKINGDAY_MONTH	Date -> Work Day for Current Month
47	WORKINGDAY_YEAR	Date -> Work Day for Current Year

Related Content

SAP Community Network

Knowledge Management

http://partner.sap.com

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.