How to Dynamically Update Broadcast Setting Variables

Applies to:

SAP NetWeaver 7.0

Summary

There may be instances, where we have a report to be broadcasted to users on daily / Monthly basis and for this we may have to change the calday / calmonth, or any other variable values every time we broadcast it, even though the broadcast is scheduled automatically. This How to document explains how this updating of the variable value can be automated without having to change it every time. Also by enhancing the code given we can even make broadcast in bursts.

Authors: M. Mohamed Abdul Rahman Shafi, Suvarna Vandana

Company: Infosys Technologies Limited.

Created on: 30 July 2010

Author Bio

Shafi and Suvarna are SAP BI Consultants working in Infosys Technologies Limited currently with around 3 years of experience. They have extensively worked in SAP BI Developments Projects.Their core skills include modelling, reporting and planning using SAPBI 3.5, SAPBI 7.0 and SAP BI Integrated Planning.

Table of Contents

Scenario

There can be many scenarios where you need to broadcast the same report / Work book with a different variable values to a set of users. Some of the possible scenarios are :

1. **Daily sales report**: Need to broadcast the daily sales report with the calmonth variable changing based on the day on which the report is being executed.
2. **Monthly Profit and loss report:** Need to broadcast Profit and Loss reports on a monthly basis where we have a selection to choose the reporting month.
3. **Forecast Report :** Need to broadcast forecasted data in comparison with different target version , where the target version is selected based on the current month of execution.

Generally in these case we create a report with Customer Exit variables (i_step = 2) for calday / calmonth objects which automatically populates values based on current calendar day.

But there are cases in which we need to broadcast existing reports which has Manual Input variable for these objects and we face with the following issue when creating a broadcast Settings.

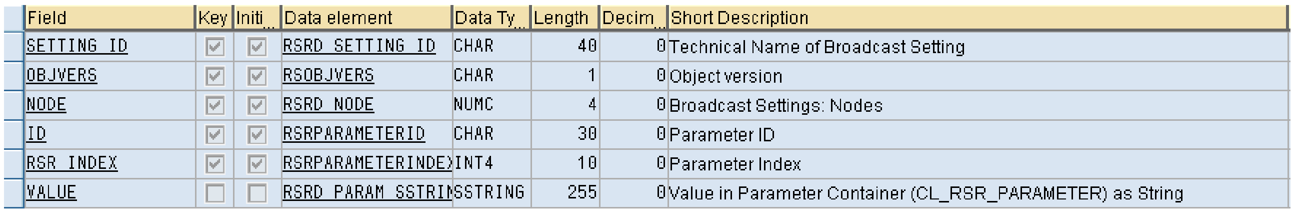
1. Since the Variable is Manual input we need to choose the Value in the Broadcast Settings.
2. Even though there is a customer Exit (i_step = 1) associated with it, this will not work when the report / Work book is being broadcasted. The reason being the variable values are populated from broadcast setting table and not from customer Exit incase of Manual Input variables.

In this case mostly, we copy the report to be broadcasted and change the Manual Input variables to customer exit variables (i_step = 2) and write the logic of population of these variables in the customer exit.

A better way of doing this will be to update the broadcast setting table with values required using a customized program as described below.

Background Info

The parameters entered in the Broadcast Settings are maintained in a table called RSRD_SETT_NODE_A, which has the following structure.



Field	Key	Initi	Data element	Data Ty	Length	Decim	Short Description
SETTING_ID	✓	✓	RSRD_SETTING_ID	CHAR	40	0	Technical Name of Broadcast Setting
OBJVERS	✓	✓	RSOBJVERS	CHAR	1	0	Object version
NODE	✓	✓	RSRD_NODE	NUMC	4	0	Broadcast Settings: Nodes
ID	✓	✓	RSRPARAMETERID	CHAR	30	0	Parameter ID
RSR_INDEX	✓	✓	RSRPARAMETERINDEX	INT4	10	0	Parameter Index
VALUE	☐	☐	RSRD_PARAM_SSTRIN	SSTRING	255	0	Value in Parameter Container (CL_RSR_PARAMETER) as String

Here there is a unique entry created for every setting we create and the details of the variable being stored as sequence of strings under the value field.

Even though we select the values for the variables in the selection screen in the broadcaster, or it is being populated with a customer exit (i_step = 1) the values are hardcoded in this table and when the broadcast is being done by the precalculation server it picks the variable value from this table and not via the i_step = 1 variable exit.

This happens only for manual entry variables. Hence the only possible option to update this variable values if the settings has to be broadcasted on periodic basis is to manually update the value in the tables.

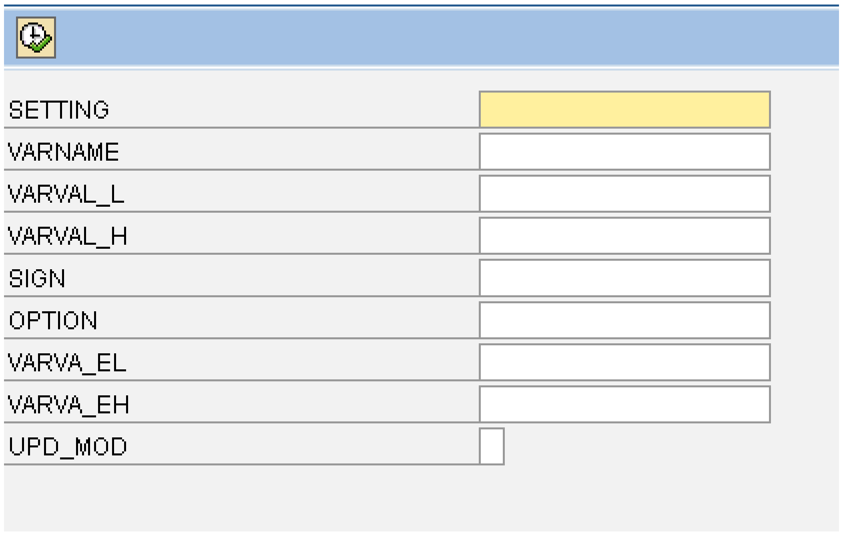
For this we need an ABAP program that can do the job which is generalized enough to be used with most of the broadcast settings.

This generalized piece of Code has to be executed Via Process chain scheduled before the broadcast settings are scheduled.

This way we make sure the settings run for the correct variable values.

How it Works

The program takes the following entries as Input.



SETTING	
VARNAME	
VARVAL_L	
VARVAL_H	
SIGN	
OPTION	
VARVA_EL	
VARVA_EH	
UPD_MOD	☐

SETTING	Technical name of the Broadcast Setting
VARNAME	Variable Name for which the value need to be changed For time characteristics it is in SAP format Ex 01.01.2010 is as 20100101
VARVAL_L	Variable Value
VARVAL_H	Variable Value Upper limit (only if the variable is of type interval)
SIGN	Sign (only if the variable is of type interval) ex. I or E
OPTION	Option (only if the variable is of type interval) ex. BT, LE,GE ...
VARVA_EL	This is a special field used by the broadcaster. For time characteristics it is in Normal format with '.' Replace by %2e. Ex 01.01.2010 is as 01%2e01%2e2010 For others it is same as the value for VARVAL_L Note : This field is used by the Broadcast Settings editor to set the internal variable value of 20100101 as 01.01.2010 in the variable Popup screen.
VARVA_EH	This is a special field used by the broadcaster. For time characteristics it is in Normal format with '.' Replace by %2e.

	Ex 01.01.2010 is as 01%2e01%2e2010 (only if the variable is of type interval) For others it is same as the value for VARVAL_H Note : This field is used by the Broadcast Settings editor to set the internal variable value of 20100101 as 01.01.2010 in the variable Popup screen.
UPD_MOD	This is a field that takes the value as either X or blank. When we pass no value to it, the program updates the given variable value in the broadcast settings. When we pass on value 'X' to it, the Program displays the values of all the parameters mentioned above from the broadcast setting.

How to Use

The code in the Appendix is a generalized code which will transform the supplied data for the variable into the RSRD_SETT_NODE_A table format.

We must write our own code or Function Module to implement the logic to update the values based on some condition and passes the value to this code by calling it in the customized code.

Let us take an example of updating the calendar month variable (TESTCALMTH) in a Broadcast setting (TESTSET).

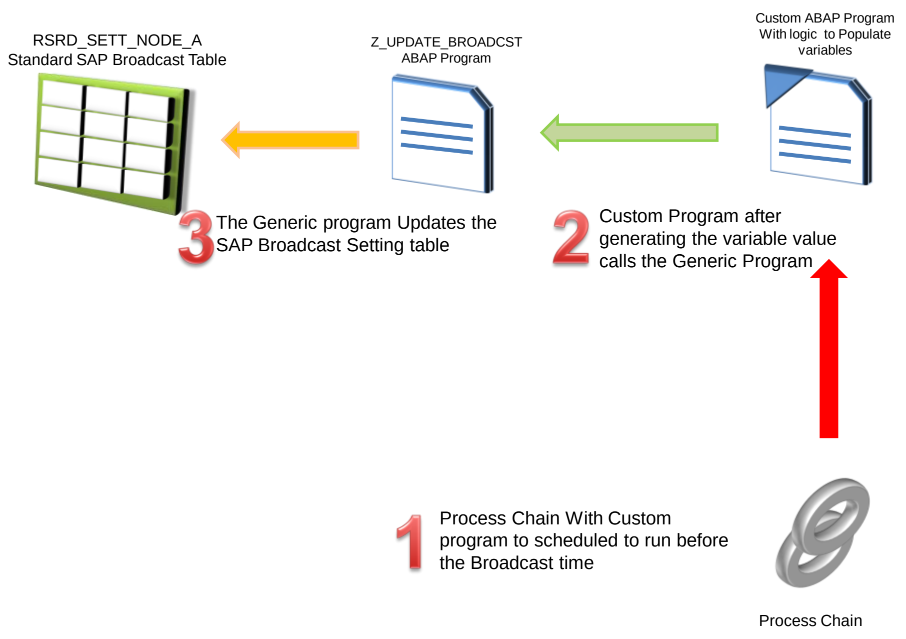
Then we write a Code / Function Module as below.

```
Data : month type c length 6,
          month_ext type c length 9.
month  = sy-datum(6).
concatenate month+4(2) ‘%2e’ month(4) into Month_ext.
Submit Z_UPDATE_BROADCST with SETTING =  ‘TESTSET’
                                            with VARNAME = ' TESTCALMTH '
                                            with VARVAL_L = month
                                            with VARVAL_H = ''
                                            with SIGN = ''
                                            with OPTION = ''
                                            with VARVA_EL = month_ext
                                            with VARVA_EH = ''
                                            with UPD_MOD = '' .
```

This code / Function Module can be included in a Process chain that is scheduled to run before the Broadcast Settings.

This way no matter when the broadcast Settings run, it always run with the updated values.

Overall Flow Diagram

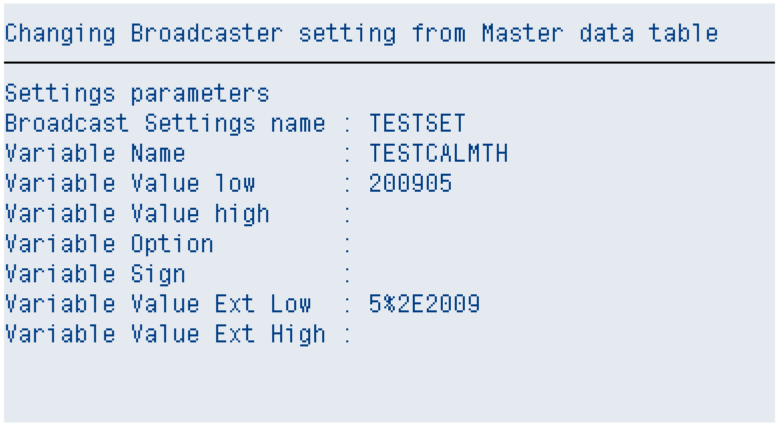

RSRD_SETT_NODE_A
Standard SAP Broadcast Table
Z_UPDATE_BROADCST
ABAP Program
Custom ABAP Program
With logic to Populate
variables
3 The Generic program Updates the SAP Broadcast Setting table
2 Custom Program after generating the variable value calls the Generic Program
1 Process Chain With Custom program to scheduled to run before the Broadcast time
Process Chain

Knowing What Value to Pass

Just run the program Z_UPDATE_BROADCST with setting, varname and upd_mod set to ‘X’.

We will get a screen with the values already present in the broadcast settings table.

This will help the developer to understand the format of the values that he must pass to the program when he writes a custom program .



```
Changing Broadcaster setting from Master data table

Settings parameters
Broadcast Settings name : TESTSET
Variable Name           : TESTCALMTH
Variable Value low      : 200905
Variable Value high     :
Variable Option         :
Variable Sign           :
Variable Value Ext Low  : 5%2E2009
Variable Value Ext High :
```

Appendix

```
REPORT  Z_UPDATE_BROADCST.


* Declaring Variables


DATA : lv_input_str TYPE STRING,
       lv_output_str TYPE STRING,
       lv_ind like sy-tabix,
       lv_out_tab_ind like sy-tabix,
       lv_out_str_len like sy-tabix,
       count(2) type c,
       lv_indx like sy-tabix,
       lv_indmax like sy-tabix,
       varo(2) type c,
       lv_var1 TYPE STRING,
       lv_brdcst type string.

* Declaring structures


TYPES: begin of t_param,
       objnm TYPE c LENGTH 40,
       varnm TYPE c LENGTH 40,
       param_nm TYPE string,
       param_val TYPE string,
      end of t_param,

* Structure for Input parameters

      begin of t_broad_param,
        broadcst type string,
        varnm TYPE string,
        varv TYPE string,
        varvh TYPE string,
        sign TYPE string,
        opt TYPE string,
        varvext type string,
        varvexth type string,
      end of t_broad_param,

* Structure for storing the parameter name and values from broadcast settings

       begin of lt_param,
       param_nm type string,
       param_val type string,
      end of lt_param,

*structure fro storing the parameter string

       begin of t_result,
       result TYPE c LENGTH 100,
      end of t_result.

* Declaring Internal tables

DATA :  t_result_value TYPE STANDARD TABLE OF t_result,
        wa_result_value TYPE t_result,
        t_param_value TYPE STANDARD TABLE OF t_param,
        wa_param_value TYPE t_param,
        lt_param_value TYPE STANDARD TABLE OF lt_param,
        w_param_value TYPE lt_param,
        t_rsrd TYPE STANDARD TABLE OF RSRD_SETT_NODE_A,
        wa_rsrd TYPE RSRD_SETT_NODE_A,
        wa_broadcast type t_broad_param.


* Input parameters Declaration

parameters : Setting type c length 20,
varname type c length 20,
varval_l type c length 20,
varval_h type c length 20,
sign type c length 20,
option type c length 20,
varva_el type c length 20,
varva_eh type c length 20,
Upd_mod(1) type c.



* Assigning variable values
lv_ind = 1.
lv_brdcst = Setting.
wa_broadcast-broadcst = Setting.
wa_broadcast-varnm = varname.
wa_broadcast-varv = varval_l.
wa_broadcast-varvh = varval_h.
wa_broadcast-sign = sign.
wa_broadcast-opt = option.
wa_broadcast-varvext =  varva_el.
wa_broadcast-varvexth = varva_eh.

*Select Broadcast Settings parameters

select * from rsrd_sett_node_a into table t_rsrd
where setting_id = lv_brdcst
and id  = 'PR_VARIABLE_STRING'
and objvers = 'A'.

describe table t_rsrd lines lv_indmax.

*get the variable values and its parameters into a variable

loop at t_rsrd into wa_rsrd.
  concatenate lv_input_str wa_rsrd-value into lv_input_str.
endloop.

*Split the parameters into an internal table as rows

SPLIT lv_input_str AT '&' INTO TABLE t_result_value IN CHARACTER MODE.

*Split the parameters into an internal table as columns

LOOP AT t_result_value INTO wa_result_value.
  SPLIT wa_result_value-result AT '=' INTO  w_param_value-param_nm  w_param_value-
param_val.


  append w_param_value to lt_param_value.
endloop.

If upd_mod ne 'X'.
```

```
* Loop at the variable that needs to be changed by reading the parameters from teh
broadcast settings


  read table lt_param_value into w_param_value
  with key param_val = wa_broadcast-varnm.

  if sy-subrc eq 0.
    count = w_param_value-param_nm+9.
    SHIFT count LEFT DELETING LEADING '0'.
    if wa_broadcast-opt eq ''.
*update different parameters using the varo as indicator
      varo = 'L'.
      concatenate 'VAR_VALUE_' count into lv_var1.
      perform change.
      varo = 'EL'.
      concatenate 'VAR_VALUE_EXT_' count into lv_var1.
      perform change.

    else.
      varo = 'L'.
      concatenate 'VAR_VALUE_LOW_' count into lv_var1.
      perform change.
      varo = 'EL'.
      concatenate 'VAR_VALUE_LOW_EXT_' count into lv_var1.
      perform change.
      varo = 'H'.
      concatenate 'VAR_VALUE_HIGH_' count into lv_var1.
      perform change.
      varo = 'EH'.
      concatenate 'VAR_VALUE_HIGH_EXT_' count into lv_var1.
      perform change.
      varo = 'O'.
      concatenate 'VAR_OPERATOR_' count into lv_var1.
      perform change.


    endif.
    varo = 'S'.
    concatenate 'VAR_SIGN_' count into lv_var1.
    perform change.


  endif.


  describe table lt_param_value lines lv_indx.

  Loop at lt_param_value into w_param_value.
    if lv_indx = sy-tabix.
      concatenate lv_output_str w_param_value-param_nm '=' w_param_value-param_val into
lv_output_str.
    else.
      concatenate lv_output_str w_param_value-param_nm '=' w_param_value-param_val '&'
into lv_output_str.
    endif.
  endloop.

  lv_out_str_len = 0.
  lv_out_tab_ind = 0.



  loop at t_rsrd into wa_rsrd.
```

```
    if sy-tabix eq lv_indmax.
      wa_rsrd-value = lv_output_str+lv_out_str_len.
    else.
      wa_rsrd-value = lv_output_str+lv_out_str_len(255).
    endif.

    modify t_rsrd from wa_rsrd transporting value where SETTING_ID = lv_brdcst AND  ID =
'PR_VARIABLE_STRING'
      and OBJVERS = 'A' and  RSR_INDEX = lv_out_tab_ind.
    .
    MODIFY RSRD_SETT_NODE_A FROM table t_rsrd.

    lv_out_str_len = sy-tabix * 255.


    lv_out_tab_ind = sy-tabix * 1000.

  endloop.

*generate status message

if sy-subrc eq 0.

  MESSAGE 'Success' TYPE 'S'.
else.
  MESSAGE 'Failed' TYPE 'E'.

endif.


else.
  Write 'Settings parameters'.
  perform varup.
endif.



*&---------------------------------------------------------------------*
*&      Form  change
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*

* Update variable values in internal table

form change.

  read table lt_param_value into w_param_value
          with key param_nm = lv_var1.
  lv_ind = sy-tabix.
  if sy-subrc eq 0.
    if varo = 'L' AND wa_broadcast-varv <> ''.
      w_param_value-param_val = wa_broadcast-varv.
      perform varchange.
    elseif varo = 'H' AND wa_broadcast-varvh <> ''.
      w_param_value-param_val = wa_broadcast-varvh.
      perform varchange.
    elseif varo = 'O' AND wa_broadcast-opt <> ''.
      w_param_value-param_val = wa_broadcast-opt.
      perform varchange.
    elseif varo = 'S' AND wa_broadcast-sign <> ''.
      w_param_value-param_val = wa_broadcast-sign.
      perform varchange.
    elseif varo = 'EL' AND wa_broadcast-varvext <> ''.
      w_param_value-param_val = wa_broadcast-varvext.
      perform varchange.
    elseif varo = 'EH' AND wa_broadcast-varvexth <> ''.
      w_param_value-param_val = wa_broadcast-varvexth.
      perform varchange.

    endif.
  endif.


endform.                    "change


*&---------------------------------------------------------------------*
*&      Form  varchange
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*

* Update variable values in broadcst settings

form varchange.
  SHIFT w_param_value-param_val LEFT DELETING LEADING '0'.
  modify lt_param_value index lv_ind from w_param_value transporting param_val.
endform.                    "varchange



*&---------------------------------------------------------------------*
*&      Form  varup
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*

* Update parameter settings in the screen

form varup.
  if varname <> ''.
    read table lt_param_value into w_param_value
      with key param_val = varname.
    if sy-subrc eq 0.
      count = w_param_value-param_nm+9.
      SHIFT count LEFT DELETING LEADING '0'.

      wa_broadcast-broadcst = lv_brdcst.
      wa_broadcast-varnm = varname.



      varo = 'L'.
      concatenate 'VAR_VALUE_' count into lv_var1.
      perform readvar.
      varo = 'EL'.
      concatenate 'VAR_VALUE_EXT_' count into lv_var1.
      perform readvar.
      varo = 'L'.
      concatenate 'VAR_VALUE_LOW_' count into lv_var1.
      perform readvar.
      varo = 'EL'.
      concatenate 'VAR_VALUE_LOW_EXT_' count into lv_var1.
      perform readvar.
      varo = 'H'.
      concatenate 'VAR_VALUE_HIGH_' count into lv_var1.
      perform readvar.
      varo = 'EH'.
      concatenate 'VAR_VALUE_HIGH_EXT_' count into lv_var1.
```

```
        perform readvar.
        varo = 'O'.
        concatenate 'VAR_OPERATOR_' count into lv_var1.
        perform readvar.
        varo = 'S'.
        concatenate 'VAR_OPERATOR_' count into lv_var1.
        perform readvar.
      endif.

      write: / 'Broadcast Settings name :',wa_broadcast-broadcst.
      write: / 'Variable Name           :',wa_broadcast-varnm.
      write: / 'Variable Value low      :',wa_broadcast-varv.
      write: / 'Variable Value high     :',wa_broadcast-varvh.
      write: / 'Variable Option         :',wa_broadcast-opt.
      write: / 'Variable Sign           :',wa_broadcast-sign.
      write: / 'Variable Value Ext Low  :',wa_broadcast-varvext.
      write: / 'Variable Value Ext High :',wa_broadcast-varvexth.


    else.
      MESSAGE 'Varname cannot be Empty' TYPE 'E'.
    endif.
  endform.                    "varup

*&---------------------------------------------------------------------*
*&      Form  readvar
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*

* read parameter values from broadcast settings

form readvar.
  read table lt_param_value into w_param_value
    with key param_nm = lv_var1.
  if sy-subrc eq 0.
    if varo = 'L'.
      wa_broadcast-varv = w_param_value-param_val.
    elseif varo = 'H'.
      wa_broadcast-varvh = w_param_value-param_val.
    elseif varo = 'S'.
      wa_broadcast-sign = w_param_value-param_val.
    elseif varo = 'O'.
      wa_broadcast-opt = w_param_value-param_val.
    elseif varo = 'EL'.
      wa_broadcast-varvext = w_param_value-param_val.
    elseif varo = 'EH'.
      wa_broadcast-varvexth = w_param_value-param_val.
    endif.

  endif.
endform.                    "readvar
```

Disclaimer and Liability Notice