Managing BEx Variable EXIT without CMOD Changes

Applies to:

SAP NW2004s (BI 7.0) SAP NW2004 (BW 3.5).For more information, visit the Business Intelligence homepage.

Summary

This article explains a different approach to manage BEx Variable with Customer Exit, with a limited coding in Include ZXRSRU01 (CMOD Enhancement RSR00001 – EXIT_SAPLRRS0_001). When many Project Teams are developing in the same time or corrections have to be deployed concurrently some conflicts may arise due to the fact that changes are locked in different Change Requests or Transports cannot be imported. This article suggests a different approach that has the purpose to avoid these conflicts.

Author: Gianfranco Vallese

Company: BGP Management Consulting S.p.A.

Created on: 24 May 2010

Author Bio

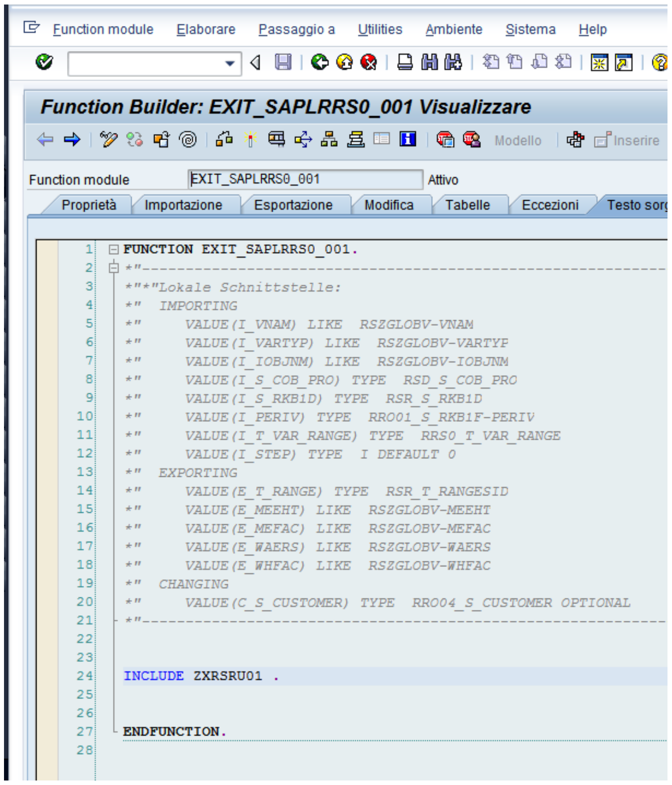
Gianfranco Vallese is Project Leader at BGP Management Consulting S.p.A..

Table of Contents

Business Scenario

SAP delivered source system user-exits allow us to calculate values for BEx Variables with processing type “Customer Exit”.

With CMOD you can activate Enhancement RSR00001 “BW: Enhancements for global variables in reporting”. In Function Module EXIT_SAPLRRS0_001 you have to write code within Include ZXRSRU01.



```
FUNCTION EXIT_SAPLRRS0_001.
*"----------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"  IMPORTING
*"     VALUE(I_VNAM) LIKE  RSZGLOBV-VNAM
*"     VALUE(I_VARTYP) LIKE  RSZGLOBV-VARTYP
*"     VALUE(I_IOBJNM) LIKE  RSZGLOBV-IOBJNM
*"     VALUE(I_S_COB_PRO) TYPE  RSD_S_COB_PRO
*"     VALUE(I_S_RKB1D) TYPE  RSR_S_RKB1D
*"     VALUE(I_PERIV) TYPE  RRO01_S_RKB1F-PERIV
*"     VALUE(I_T_VAR_RANGE) TYPE  RRS0_T_VAR_RANGE
*"     VALUE(I_STEP) TYPE  I DEFAULT 0
*"  EXPORTING
*"     VALUE(E_T_RANGE) TYPE  RSR_T_RANGESID
*"     VALUE(E_MEEHT) LIKE  RSZGLOBV-MEEHT
*"     VALUE(E_MEFAC) LIKE  RSZGLOBV-MEFAC
*"     VALUE(E_WAERS) LIKE  RSZGLOBV-WAERS
*"     VALUE(E_WHFAC) LIKE  RSZGLOBV-WHFAC
*"  CHANGING
*"     VALUE(C_S_CUSTOMER) TYPE  RRO04_S_CUSTOMER OPTIONAL
*"----------------------------------------------------------------


INCLUDE ZXRSRU01 .


ENDFUNCTION.
```

Picture 1: EXIT_SAPLRRS0_001

Generally Include ZXRSRU01 is divided in four main sections, each corresponding to a different value of variable I_STEP (see OSS Note 492504 – Dependent customer exit-type variables).

I_STEP = 0	The enhancement is not called from the BEx Query variable screen. The call can come from the authorization check or from the Monitor.
I_STEP = 1	The first step is before the processing of the variable pop-up and gets called for every variable of the processing type customer exit. You can use this step to fill your variable with default values.
I_STEP = 2	The second step is called after the processing of the variable pop-up. This step is called only for those variables that are not marked as ready for input and are set to mandatory variable entry
I_STEP = 3	The third step (I_STEP = 3) is called after all variable processing and gets called only once and not per variable. Here you can validate the user entries.

Within each I_STEP value generally you code each variable in the following way:

```
CASE i_STEP.
  WHEN '0'.
    ...

  WHEN '1'.
    ...

  WHEN '2'.
    CASE i_vnam.
      WHEN 'VARIABLE1'.
        LOOP AT i_t_var_range INTO loc_var_range
          WHERE vnam = 'XYZ'.
          ...

        ENDLOOP.
      WHEN 'VARIABLE2'.
        ...

      WHEN 'VARIABLE3'.
        ...

    ENDCASE.

  WHEN '3'.
    ...

ENDCASE.
```

It is quite common having more than one thousand rows of ABAP code in the same Include, written by different developers.

Transport management can be challenging especially when you have large number of BEx Variables and different people or teams are working on the user-exits. Object may be locked by someone else while you are trying to incorporate an urgent fix to resolve a production issue. This leads to reversal of code and retesting of the variables / code which are redundant and unproductive.

Similar problems can arise in the back end CMOD implementations (DataSources Enhancements): SAP delivered an How To (see Related Content) named “How To ... Dynamically Call DataSource Specific Programs in Source System User-exits” that had the purpouse to solve the problem. On the front end side Enhancements you can engage the same problems, that can be solved with the approach proposed in this document.

Step by Step Procedure

The main logic here proposed is having a dynamic CALL FUNCTION to a different Function Module for each BEx Variable. The Function Module to be associated to each Variable is stored in a Custom Table that can be directly updated using a view / transports or custom program.

No additional transport of Include ZXRSRU01 will be necessary: individual Project Teams will responsible for their own Function Groups / Functions Modules and relative table entries.

This solution can be implemented for a large number of variables: all those associated to I_STEP 0, 1, and 2.

In the following paragraphs you can see the details concerning this implementation:

- Custom Function Group / Function Modules

Custom Table

- CMOD Include ZXRSRU01

Custom Function Group / Function Modules

With transaction SE37 - ABAP Function Modules create a new Function Group.

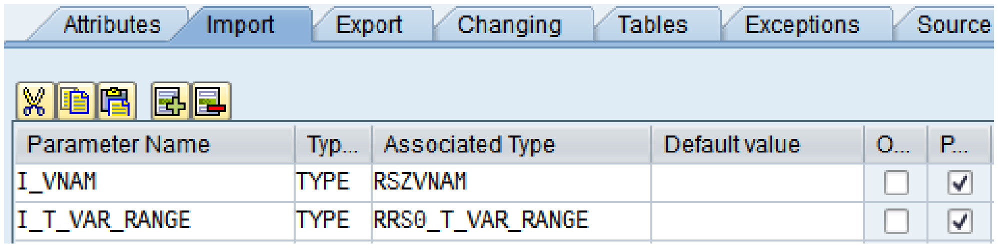
In Global Data make the following declarations

```
FUNCTION-POOL ZBEXVAR_001.                       "MESSAGE-ID ..

TYPE-POOLS: RSR,
            RRS0.

* wa used to Loop over I_T_VAR_RANGE
DATA: loc_var_range TYPE RRS0_S_VAR_RANGE.

* wa for appending E_T_RANGE
DATA: l_s_range TYPE rsr_s_rangesid.
```

It makes sense to have a separate Function Group for each Project / Team, in order to avoid concurrent transport problems.

Then you have to code a single function Module for each variable you need to process. Function Modules have to meet the following requirements concerning importing and exporting parameters.

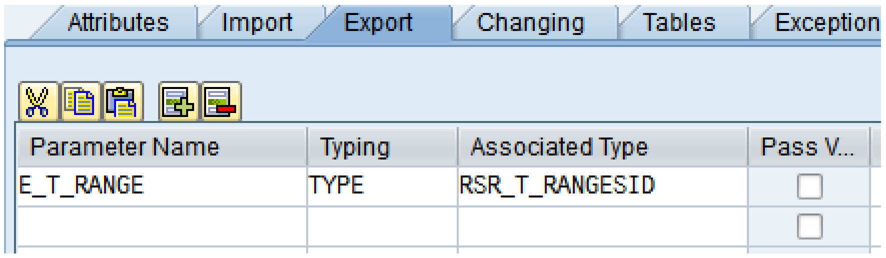
- Importing Parameters



Parameter Name	Typ...	Associated Type	Default value	O...	P...
I_VNAM	TYPE	RSZVNAM		☐	✓
I_T_VAR_RANGE	TYPE	RRS0_T_VAR_RANGE		☐	✓

Parameter	Type	Notes
I_VNAM	RSZVNAM	Technical name Variable
I_T_VAR_RANGE	RRS0_T_VAR_RANGE	Internal Table with all variable values

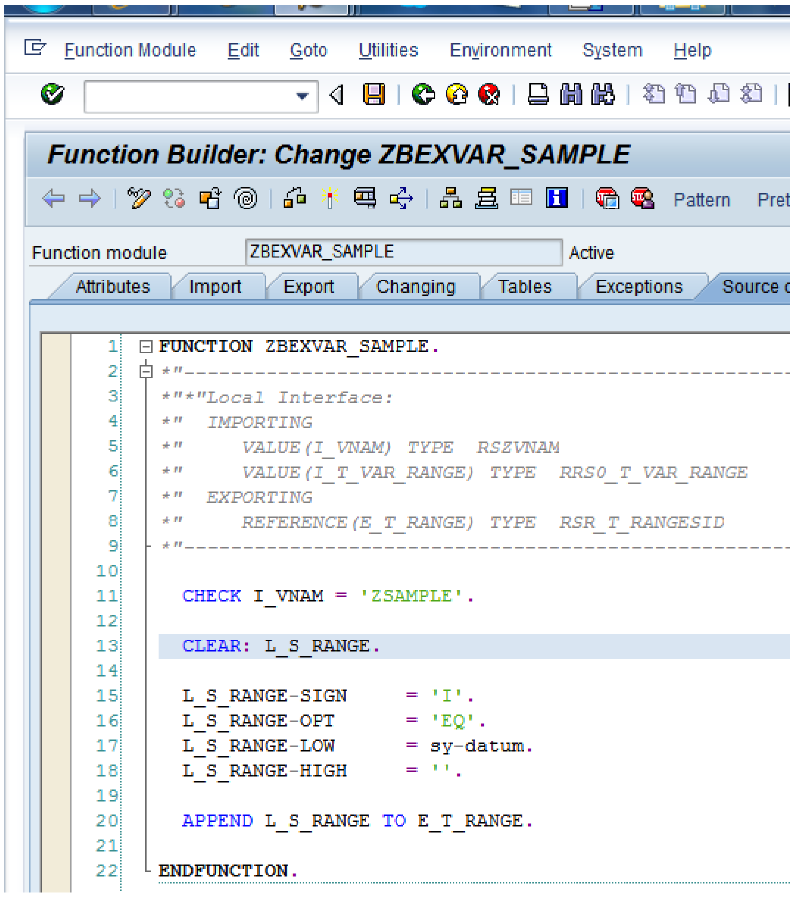
- Exporting Parameters



Parameter Name	Typing	Associated Type	Pass V...
E_T_RANGE	TYPE	RSR_T_RANGESID	☐
			☐

Parameter	Type	Notes
E_T_RANGE	RSR_T_RANGESID	Internal table with new variable values

Within each Function Module you can code the same logics you were used to place in Include ZXRSRU01.



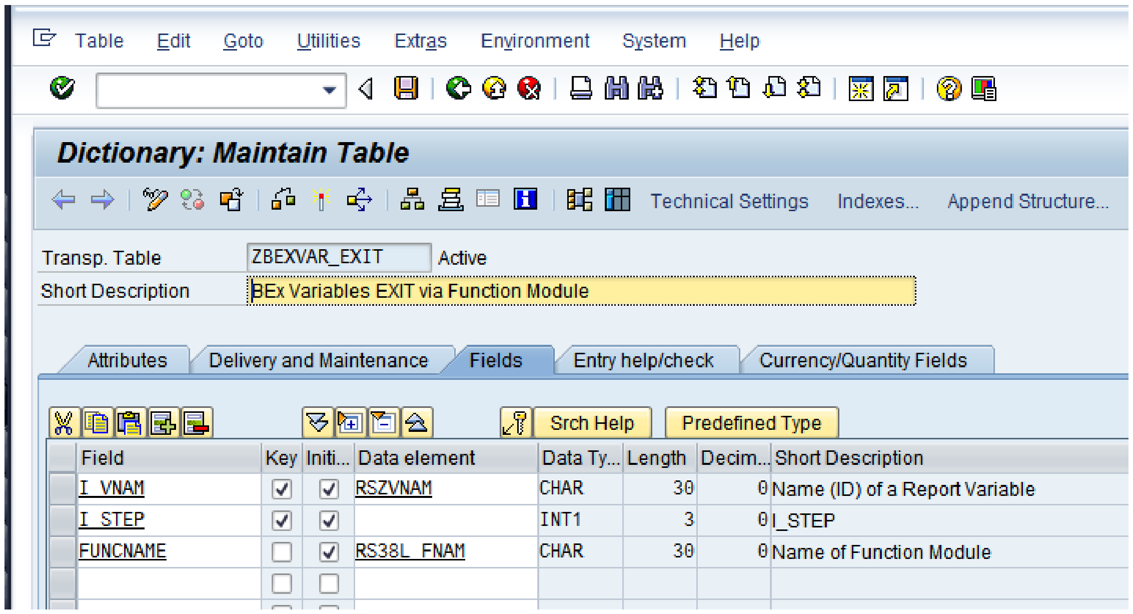
```
FUNCTION ZBEXVAR_SAMPLE.
*"----------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(I_VNAM) TYPE  RSZVNAM
*"     VALUE(I_T_VAR_RANGE) TYPE  RRS0_T_VAR_RANGE
*"  EXPORTING
*"     REFERENCE(E_T_RANGE) TYPE  RSR_T_RANGESID
*"----------------------------------------------------------

  CHECK I_VNAM = 'ZSAMPLE'.

  CLEAR: L_S_RANGE.

  L_S_RANGE-SIGN      = 'I'.
  L_S_RANGE-OPT       = 'EQ'.
  L_S_RANGE-LOW       = sy-datum.
  L_S_RANGE-HIGH      = ''.

  APPEND L_S_RANGE TO E_T_RANGE.

ENDFUNCTION.
```

Picture 2: Sample Function Module

Custom Table

A Custom table (ZBEXVAR_EXIT) will be used to link a specific Function Module for each BEx Variable that must be processed within CMOD Enhancement.

With transaction SE11 – ABAP Dictionary Maintenance create a new table, named ZBEXVAR_EXIT with the fields shown in the following picture:



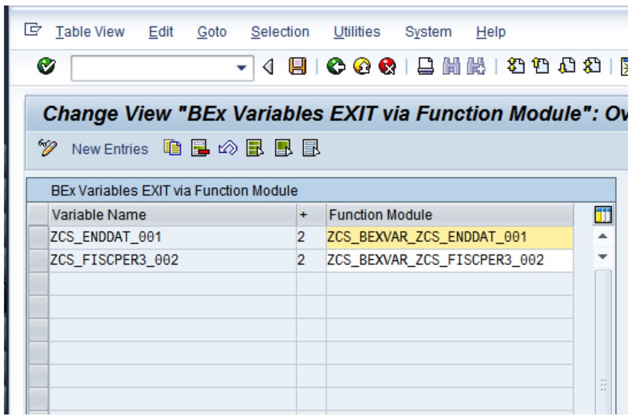
Transp. Table ZBEXVAR_EXIT Active
Short Description BEx Variables EXIT via Function Module

Field	Key	Initi...	Data element	Data Ty...	Length	Decim...	Short Description
I_VNAM	✓	✓	RSZVNAM	CHAR	30	0	Name (ID) of a Report Variable
I_STEP	✓	✓		INT1	3	0	I_STEP
FUNCNAME		✓	RS38L_FNAM	CHAR	30	0	Name of Function Module

Picture 3: Table ZBEXVAR_EXIT

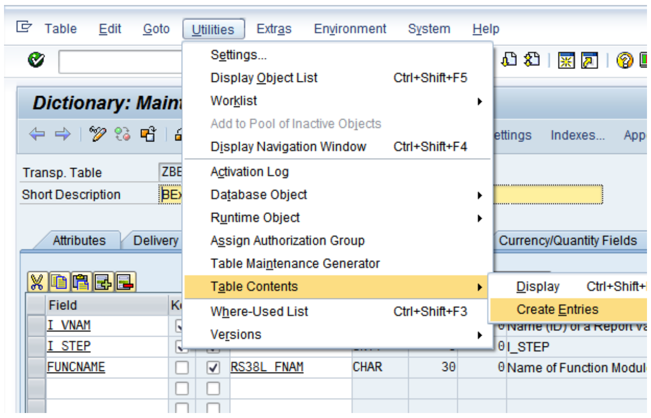
Each entry in table ZBEXVAR_EXIT corresponds to a single variable / processing step: each of these couples must be processed with a specific function module.

In order to manage table contents you can create a maintenance view or update table directly, with your Variables.



BEx Variables EXIT via Function Module

Variable Name	+	Function Module
ZCS_ENDDAT_001	2	ZCS_BEXVAR_ZCS_ENDDAT_001
ZCS_FISCPER3_002	2	ZCS_BEXVAR_ZCS_FISCPER3_002



Picture 4: Maintenance View

Picture 5: Create Table Entries

CMOD Include ZXRSRU01

Replace the code in Include ZXRSRU01 with the code explained in detail in APPENDIX I: Include ZXRSRU01 ABAP Code.

The main difference with the code generally implemented is the fact that you won't have thousands of ABAP Code lines sequentially typed in the same Include, but one dynamic CALL FUNCTION that allows you to:

- Reduce Transport Issues
- Have ABAP Objects locked by responsible developers
- Minimize future changes in Include ZXRSRU01

APPENDIX I: Include ZXRSRU01 ABAP Code

Notes — **ABAP Code**

```
*&---------------------------------------------------------------------*
*&  Include           ZXRSRU01                                         *
*&---------------------------------------------------------------------*
```

Work Area for table ZBEXVAR_EXIT

```
DATA: wa_ZBEXVAR_EXIT TYPE ZBEXVAR_EXIT.
```

Internal Tabel with FM Results

```
DATA: tb_RANGESID     TYPE RSR_T_RANGESID.
```

The same approach is used for

I_STEP = 0

```
CASE I_STEP.
  WHEN 0
* The enhancement is not called from the variable screen. The call can
* come from the authorization check or from the Monitor.
```

I_STEP = 1

```
    OR 1
* The first step (I_STEP = 1) is before the processing of the variable
* pop-up and gets called for every variable of the processing type
* customer exit. You can use this step to fill your variable with
* default values.
```

I_STEP = 2

```
    OR 2.
* The second step (I_STEP = 2) is called after the processing of the
* variable pop-up. This step is called only for those variables that
* are not marked as ready for input and are set to mandatory variable
* entry.
```

Search for a specific Function Module related to Variable I_VNAM

```
    SELECT SINGLE *
      INTO wa_ZBEXVAR_EXIT
      FROM ZBEXVAR_EXIT
     WHERE I_VNAM = I_VNAM
       AND I_STEP = I_STEP.


    CHECK sy-subrc = 0.
    TRY.
```

Execute specific Function Module

```
        CALL FUNCTION wa_ZBEXVAR_EXIT-FUNCNAME
          EXPORTING
            I_VNAM        = I_VNAM
            I_T_VAR_RANGE = I_T_VAR_RANGE
          IMPORTING
            E_T_RANGE     = tb_RANGESID.
```

Transfer Function Module Results

```
        APPEND LINES OF tb_RANGESID TO E_T_RANGE.

      CATCH CX_SY_DYN_CALL_ILLEGAL_FUNC
            CX_SY_DYN_CALL_ILLEGAL_TYPE
            CX_SY_DYN_CALL_PARAM_MISSING
            CX_SY_DYN_CALL_PARAM_NOT_FOUND.
    ENDTRY.
```

Notes	ABAP Code

I_STEP = 3 can be processed with other function Modules (one for each Project / Team) or as usual, with direct coding.
 	
Bear in mind that I_STEP = 3 is called once, and not per single variable, as other I_STEPs, so the same approach described before is not suitable
	```
	  WHEN 3.
	*The third step (I_STEP = 3) is called after all variable processing and
	* gets called only once and not per variable. Here you can validate the
	* user entries.
	    CALL FUNCTION 'ZBEXVAR_I_STEP_3'
	      EXPORTING
	        I_T_VAR_RANGE   = I_T_VAR_RANGE
	      EXCEPTIONS
	        NO_REPLACEMENT = 1
	        OTHERS         = 2.
	    IF SY-SUBRC <> 0.
	* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
	*         WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
	    ENDIF.

	ENDCASE.
	```

Since the fact that I_STEP = 3 gets called only once and not for single variables, we cannot use the same logic. A possible solution to this problem is using several function modules, one for each Project / Team, where you can implement your checks independently.

As you can see, in the sample code we have used another function module, that uses I_T_VAR_RANGE as only import parameter: it will raise NO_RELACEMENT exception if any control will be failed.

Related Content

Further information on these topics can be found using the links below:

OSS Note 492504 – Dependent customer exit-type variables

BEx Customer Exits

How to Dynamically Call DataSource Specific Programs in Source System User-Exits (NW7.0)

Variable Exit in Sap BI 7.0 - How to Start

For more information, visit the Business Intelligence homepage.

Disclaimer and Liability Notice