ABSS Europe
Advanced Business Strategies & Solutions

www.biportal.org

SAP BW Guidelines

BEx User Exit Variables Creation

by Federico Cattaneo

2013

Table of Contents

1 Introduction

The purpose of this document is to provide a guide to create BEx user exit variables. All the ABAP coding/logic should follow this procedure.

1.1 Reasons

Using the standard user exit can generate problems if multiple persons are developing at the same time. The same ABAP object can be locked and create dependencies within projects.
Also the number of lines of code will increase rapidly and it will become difficult to maintain it.

1.2 Approach

The idea is to use dynamic classes/methods. Each infoobject with BEx user exit variables will have a class, and each variable will have a method within that class.
The ABAP logic to fill the variable filter values will have to be included in this method which is going to be called dynamically.

1.3 Advantage

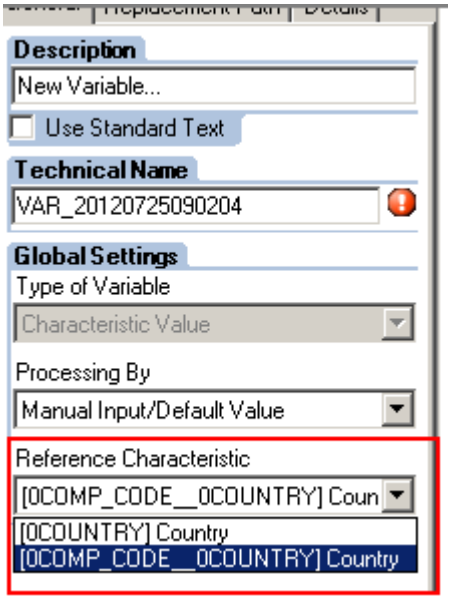
By using this approach the ABAP code will be modularized making it easy to manage and maintain.
The dependency between projects will be minimized by the creation of one class per infoobject. Of course there can be some conflicts (two projects creating variables for the same infoobject at the same time) but it is going to be more manageable.
And finally, the fact of having all variables related to the same infoobject together within the same class will make it easier to check them and possible re-use one instead of creating a new one.

2 Step-by-step procedure

2.1 Create the BEx variable

Within BEx query designer you will create a new user exit variable as you usually do. Please consider the following:

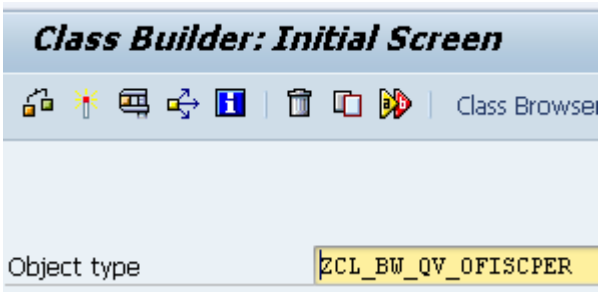
- The variable technical name should follow the naming convention, but **it should not be longer than 26 characters**
- If the variable is based on a navigational attribute **always select the base infoobject as the *reference characteristic***. By doing this it will be possible to re-use it in other infoproviders where the base characteristic is present



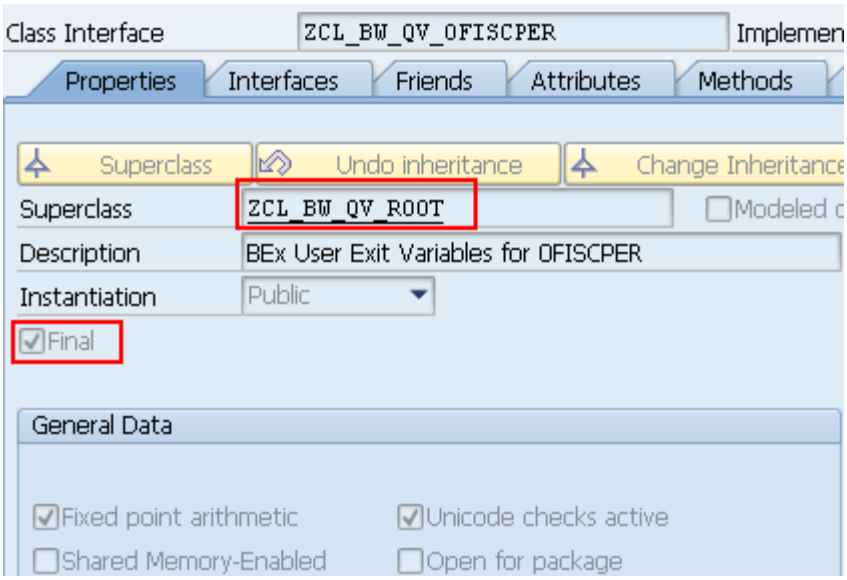
2.2 Create the class for the infoobject

There has to be one class per infoobject, with the following constrains:

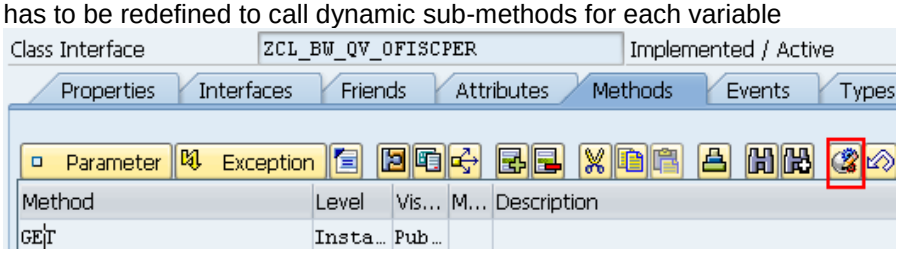
- The class name must have the format “ZCL_BW_QV_” concatenated with the complete infoobject technical name



- The class must have the class ZCL_BW_QV_ROOT as the superclass, and must be flagged as “final”



- By inheriting from class ZCL_BW_QV_ROOT a method called “GET” will be available with the same parameters existing in the standard user exit EXIT_SAPLRRS0_001. This method has to be redefined to call dynamic sub-methods for each variable



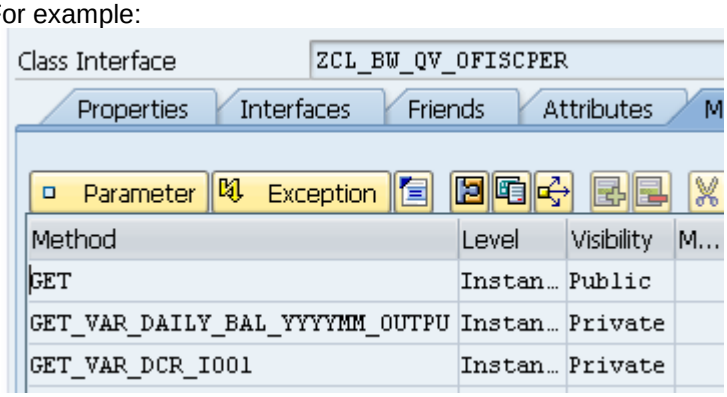
Example code for the redefinition:

```
METHOD get.

  DATA lv_prefix TYPE c LENGTH 10.
  DATA lv_method_name TYPE c LENGTH 30.


  lv_prefix = 'GET_'.
  CONCATENATE lv_prefix i_vnam INTO lv_method_name .


  TRY.
     CALL METHOD me->(lv_method_name)
       EXPORTING
         i_t_var_range = i_t_var_range
         i_step        = i_step
       CHANGING
         e_t_range     = e_t_range.

   CATCH cx_sy_create_object_error.
   CATCH cx_sy_ref_is_initial.
   CATCH cx_root.
  ENDTRY.

ENDMETHOD.
```

2.3 Create the method for the BEx variable

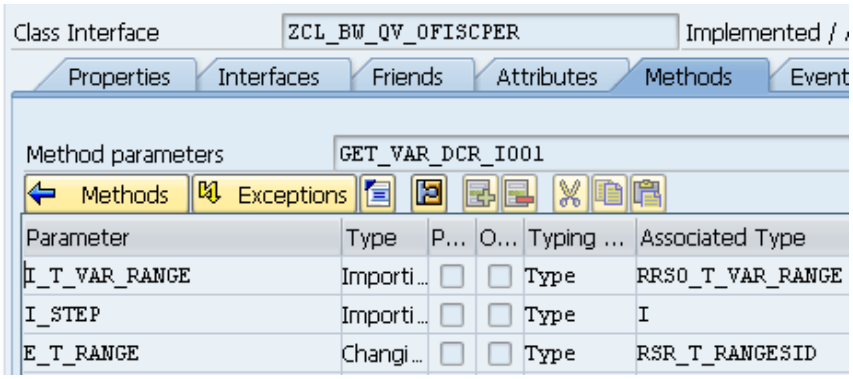
There has to be one method per each BEx variable, with the following constrains:

- The name of the method must start with “GET_” concatenated with the variable technical name. For example:



- The method visibility should be “Private”

- The description should be filled with a short explanation of the variable purpose
- The method can use any of the parameters that are available in the standard user exit. Following with the same example:



Note: If other parameters are required then the redefinition code for the method GET might need to be changed

- The *i_step* has to be considered as in the standard user exit
- Here is one example code:

```
METHOD get_var_dcr_i001.
  DATA: l_s_var_range LIKE LINE OF i_t_var_range,
        l_s_range TYPE rsr_s_rangesid,
        lv_period TYPE /bi0/oifiscper,
        lv_period2 TYPE /bi0/oifiscper,
        lv_year TYPE /bi0/oifiscper,
        lv_year2 TYPE /bi0/oifiscper.

  IF i_step EQ 2.
    READ TABLE i_t_var_range INTO l_s_var_range
      WITH KEY vnam = '0I_FYPER'.
    IF sy-subrc EQ 0.
      l_s_range-sign  ='I'.
      l_s_range-opt   ='EQ'.

      lv_period  = l_s_var_range-low+4(3).
      lv_period2 = l_s_var_range-high+4(3).

      lv_year   = l_s_var_range-low+0(4) -
1.    "YYYYPPP is the format of '0I_FYPER'
      lv_year2  = l_s_var_range-high+0(4) - 1.

      CONCATENATE lv_year lv_period INTO l_s_range-low.
      CONCATENATE lv_year2 lv_period2 INTO l_s_range-high.

      APPEND l_s_range TO e_t_range.
    ENDIF.
  ENDIF.

ENDMETHOD.
```

3 Implementation

This section will provide a guideline on what it is required to implement the previous explained logic.

3.1 New Classes

Two new classes have to be created. One will be used in the standard user exit EXIT_SAPLRRS0_001 and the other one will be the superclass used in each infoobject class.

3.1.1 Class ZCL_BW_QV_EXIT

This class will be called in the standard user exit. It will be the only code included there. Its purpose is to dynamically call a class/method according to the BEx variable processed.

Below there is an example of its code. As mentioned before, it will include the same parameters existing EXIT_SAPLRRS0_001.

```
class ZCL_BW_QV_EXIT definition
  public
  create public .

*"* public components of class ZCL_BW_QV_EXIT
*"* do not include other source files here!!!
public section.

  type-pools RSR .
  type-pools RRS0 .
  class-methods GET
    importing
      !I_VNAM type RSZVNAM
      !I_VARTYP type RSZVARTYP
      !I_IOBJNM type RSIOBJNM
      !I_S_COB_PRO type RSD_S_COB_PRO
      !I_S_RKB1D type RSR_S_RKB1D
      !I_PERIV type PERIV
      !I_T_VAR_RANGE type RRS0_T_VAR_RANGE
      !I_STEP type I default 0
    changing
      !E_T_RANGE type RSR_T_RANGESID
    exceptions
      NO_PPROCESSING .
*"* protected declarations
*"* protected components of class ZCL_BW_QV_EXIT
*"* do not include other source files here!!!
protected section.

  class-methods GET_BASIS_IOBJ_1
    importing
      !I_REF_IOBJNM type RSIOBJNM
    returning
      value(R_BASIS_IOBJNM) type RSIOBJNM .
  class-methods GET_BASIS_IOBJ_2
    importing
      !I_REF_IOBJNM type RSIOBJNM
    returning
      value(R_BASIS_IOBJNM) type RSIOBJNM .
*"* private declarations
*"* private components of class ZCL_BW_QV_EXIT
```

```
*"* do not include other source files here!!!
private section.
ENDCLASS.


CLASS ZCL_BW_QV_EXIT IMPLEMENTATION.


* <SIGNATURE>---------------------------------------------------------------------------------------+
* | Static Public Method ZCL_BW_QV_EXIT=>GET
* +---------------------------------------------------------------------------------------------------+
* | [--->] I_VNAM                         TYPE        RSZVNAM
* | [--->] I_VARTYP                       TYPE        RSZVARTYP
* | [--->] I_IOBJNM                       TYPE        RSIOBJNM
* | [--->] I_S_COB_PRO                    TYPE        RSD_S_COB_PRO
* | [--->] I_S_RKB1D                      TYPE        RSR_S_RKB1D
* | [--->] I_PERIV                        TYPE        PERIV
* | [--->] I_T_VAR_RANGE                  TYPE        RRS0_T_VAR_RANGE
* | [--->] I_STEP                         TYPE        I (default =0)
* | [<--->] E_T_RANGE                     TYPE        RSR_T_RANGESID
* | [EXC!] NO_PPROCESSING
* +---------------------------------------------------------------------------------------</SIGNATURE>
METHOD get.

** If the variable is on navigational object - then return the base object.

  DATA o_qv           TYPE REF TO zcl_bw_qv_root.
  DATA lv_basis_iobjnm TYPE rsiobjnm.
  DATA lv_prefix      TYPE c LENGTH 10.


  DATA lv_imp_class_name TYPE c LENGTH 40.

  lv_prefix = 'ZCL_BW_QV_'.

  IF i_step EQ 1 OR i_step EQ 2.
    lv_basis_iobjnm = get_basis_iobj_1( i_iobjnm ).
    CONCATENATE lv_prefix lv_basis_iobjnm INTO lv_imp_class_name .
  ELSEIF i_step EQ 3.
    CONCATENATE lv_prefix i_s_rkb1d-infocube INTO lv_imp_class_name .
  ENDIF.


  TRY.
      CREATE OBJECT o_qv TYPE (lv_imp_class_name).
      o_qv->get(
        EXPORTING
          i_vnam       = i_vnam
          i_vartyp     = i_vartyp
          i_iobjnm     = i_iobjnm
          i_s_cob_pro  = i_s_cob_pro
          i_s_rkb1d    = i_s_rkb1d
          i_periv      = i_periv
          i_t_var_range = i_t_var_range
          i_step       = i_step
        CHANGING
          e_t_range    = e_t_range ).

    CATCH cx_sy_create_object_error.
```

```
    CATCH cx_sy_ref_is_initial.
    CATCH cx_root.
  ENDTRY.

  IF o_qv IS INITIAL.

    IF i_step EQ 1 OR i_step EQ 2.
      CLEAR lv_imp_class_name.
      lv_basis_iobjnm = get_basis_iobj_2( i_iobjnm ).
      CONCATENATE lv_prefix lv_basis_iobjnm INTO lv_imp_class_name .
    ENDIF.

    TRY.
        CREATE OBJECT o_qv TYPE (lv_imp_class_name).
        o_qv->get(
          EXPORTING
            i_vnam       = i_vnam
            i_vartyp     = i_vartyp
            i_iobjnm     = lv_basis_iobjnm
            i_s_cob_pro  = i_s_cob_pro
            i_s_rkb1d    = i_s_rkb1d
            i_periv      = i_periv
            i_t_var_range = i_t_var_range
            i_step       = i_step
          CHANGING
            e_t_range    = e_t_range ).

      CATCH cx_sy_create_object_error.
      CATCH cx_sy_ref_is_initial.
      CATCH cx_root.
    ENDTRY.
  ENDIF.


ENDMETHOD.


* <SIGNATURE>---------------------------------------------------------------------------------------+
* | Static Protected Method ZCL_BW_QV_EXIT=>GET_BASIS_IOBJ_1
* +--------------------------------------------------------------------------------------------------+
* | [--->] I_REF_IOBJNM                   TYPE        RSIOBJNM
* | [<-()] R_BASIS_IOBJNM                 TYPE        RSIOBJNM
* +--------------------------------------------------------------------------------------</SIGNATURE>
METHOD get_basis_iobj_1.

  DATA l_nav_iobjnm TYPE rsiobjnm.
  DATA l_ref_iobjnm TYPE rsiobjnm.
  DATA l_basis_iobjnm TYPE rsiobjnm.

  r_basis_iobjnm = i_ref_iobjnm .
  l_nav_iobjnm = i_ref_iobjnm .
  l_ref_iobjnm =  i_ref_iobjnm .


* Check for navigational attribute.
  IF i_ref_iobjnm CS '__' .
    SELECT SINGLE attrinm INTO l_nav_iobjnm
      FROM rsdatrnav
      WHERE atrnavnm = i_ref_iobjnm AND
```

```
        objvers = 'A'.
   IF l_nav_iobjnm IS NOT INITIAL.
     l_ref_iobjnm =  l_nav_iobjnm.
   ENDIF.



   SELECT SINGLE chabasnm
     INTO  l_basis_iobjnm
     FROM  rsdcha
     WHERE chanm = l_ref_iobjnm AND
        objvers = 'A'.

** set Return value
   r_basis_iobjnm = l_basis_iobjnm .

  ENDIF.

ENDMETHOD.


* <SIGNATURE>---------------------------------------------------------------------------------------+
* | Static Protected Method ZCL_BW_QV_EXIT=>GET_BASIS_IOBJ_2
* +--------------------------------------------------------------------------------------------------+
* | [--->] I_REF_IOBJNM                   TYPE        RSIOBJNM
* | [<-()] R_BASIS_IOBJNM                 TYPE        RSIOBJNM
* +---------------------------------------------------------------------------------------</SIGNATURE>
METHOD get_basis_iobj_2.

** If the variable is on reference object - then return the base object.

  DATA l_nav_iobjnm TYPE rsiobjnm.
  DATA l_ref_iobjnm TYPE rsiobjnm.
  DATA l_basis_iobjnm TYPE rsiobjnm.

  r_basis_iobjnm = i_ref_iobjnm .
  l_ref_iobjnm =  i_ref_iobjnm.

* Check for navigational attribute.

  SELECT SINGLE chabasnm
    INTO  l_basis_iobjnm
    FROM  rsdcha
    WHERE chanm = l_ref_iobjnm AND
       objvers = 'A'.

** set Return value
  r_basis_iobjnm = l_basis_iobjnm .


ENDMETHOD.
ENDCLASS.
```

3.1.2 Class ZCL_BW_QV_ROOT

This class will be the superclass of all infoobject classes. It does not have to be flagged as *Final*.

Below there is an example of its code.

```
class ZCL_BW_QV_ROOT definition
  public
  create public .

*"* public components of class ZCL_BW_QV_ROOT
*"* do not include other source files here!!!
public section.

  type-pools RSR .
  type-pools RRS0 .
  methods GET
    importing
      !I_VNAM type RSZVNAM
      !I_VARTYP type RSZVARTYP
      !I_IOBJNM type RSIOBJNM
      !I_S_COB_PRO type RSD_S_COB_PRO
      !I_S_RKB1D type RSR_S_RKB1D
      !I_PERIV type PERIV
      !I_T_VAR_RANGE type RRS0_T_VAR_RANGE
      !I_STEP type I
    changing
      !E_T_RANGE type RSR_T_RANGESID
    exceptions
      ZCX_ECON_EMPTY_FILE .
*"* protected declarations
*"* protected components of class ZCL_BW_QV_EXIT1
*"* do not include other source files here!!!
protected section.
*"* private declarations
*"* private components of class ZCL_BW_QV_EXIT1
*"* do not include other source files here!!!
private section.
ENDCLASS.



CLASS ZCL_BW_QV_ROOT IMPLEMENTATION.


* <SIGNATURE>---------------------------------------------------------------------------------------+
* | Instance Public Method ZCL_BW_QV_ROOT->GET
* +--------------------------------------------------------------------------------------------------+
* | [--->] I_VNAM                         TYPE       RSZVNAM
* | [--->] I_VARTYP                       TYPE       RSZVARTYP
* | [--->] I_IOBJNM                       TYPE       RSIOBJNM
* | [--->] I_S_COB_PRO                    TYPE       RSD_S_COB_PRO
* | [--->] I_S_RKB1D                      TYPE       RSR_S_RKB1D
* | [--->] I_PERIV                        TYPE       PERIV
* | [--->] I_T_VAR_RANGE                  TYPE       RRS0_T_VAR_RANGE
* | [--->] I_STEP                         TYPE       I
* | [<-->] E_T_RANGE                      TYPE       RSR_T_RANGESID
* | [EXC!] ZCX_ECON_EMPTY_FILE
* +-------------------------------------------------------------------------------------</SIGNATURE>
METHOD get.

*
*  DATA lv_prefix TYPE c LENGTH 10.
```

```
*  DATA lv_method_name TYPE c LENGTH 60.
*
*  lv_prefix = 'GET_'.
*
*  CONCATENATE lv_prefix i_vnam INTO lv_method_name .
*
*
*  try.
*
*     call method me->(lv_method_name)
*          EXPORTING
*            i_t_var_range = i_t_var_range
*          CHANGING
*            e_t_range     = e_t_range .
*
*   catch cx_sy_create_object_error.
*   catch cx_sy_ref_is_initial.
*   catch cx_root.
*  endtry.



ENDMETHOD.
ENDCLASS.
```

3.2 User Exit EXIT_SAPLRRS0_001

In this user exit the only code that will be included is the call to the method that will process and call each specific class according to the infoobject. Once transported there will be no need to make changes to it.

Below there is an example of its code.

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


TRY.
   zcl_bw_qv_exit=>get(
     EXPORTING
       i_vnam = i_vnam
       i_vartyp = i_vartyp
       i_iobjnm = i_iobjnm
       i_s_cob_pro = i_s_cob_pro
       i_s_rkb1d = i_s_rkb1d
       i_periv = i_periv
       i_t_var_range = i_t_var_range
       i_step = i_step
     CHANGING
       e_t_range     = e_t_range ).

*  CATCH no_processing.
  CATCH cx_sy_create_object_error.
  CATCH cx_sy_ref_is_initial.
  CATCH cx_root.
ENDTRY.
```