

How-To Derive a BPS Variable for all users from a BPS User Specific Variable Value

1)	Business Scenario	2
2)	System Requirements.....	2
3)	Step – By – Step Solution	2
4)	Appendix A: Function Module “Z_BPS_DERIVE_VAR”	7

Disclaimer:

The Author is not an employee or representative of SAP. Any solution provided by The Author is neither supported nor endorsed by SAP.

These materials are provided “as is” without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

The Author shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

The Author does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. The Author has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

The Author does not warrant the correctness and completeness of the Code given herein, and The Author shall not be liable for errors or damages caused by the usage of the Code.

These materials are subject to change without notice.

1) Business Scenario

This document contains information on how to fill a fixed value characteristic variable from a user input or user restricted variable value. In essence, this gives an end user control over a “global” variable value. A sample use for this feature is an administrator would like to select a value in his input screen (such as the current plan version, or plan year) and have this value apply to all planning users.

2) System Requirements

This solution has been tested on BW BPS versions 3.1 - 3.5, however it should theoretically work for previous versions of SEM BPS and BW BPS.

3) Step – By – Step Solution

a) Create the ABAP function “Z_BPS_DERIVE_VAR”

Copy the function TEMPLATE_EXIT (in function group UPFX) as a template for the import and export parameters. Replace the source code with the source code in Appendix A.

b) Create the data elements for the source and target BPS Variables

Create the following data elements:

Data element	Z_BPS_SRC_VAR	Active
Short Text	BPS Source Variable	
<div style="display: flex; justify-content: space-around; border-bottom: 1px solid black;"> Attributes Data Type Further Characteristics Field Label </div>		
<input checked="" type="radio"/> Elementary Type <input checked="" type="radio"/> Domain		
	UPC VARIABLE	<input checked="" type="checkbox"/> Variable
Data Type	CHAR	Character String
Length	8	Decimal Places 0

Data element	Z_BPS_TAR_VAR	Active
Short Text	BPS Target Variable	

Attributes	Data Type	Further Characteristics	Field Label
<input checked="" type="radio"/> Elementary Type <input type="radio"/> Domain			
		UPC VARIABLE	<input checked="" type="checkbox"/> Variable
Data Type	CHAR	Character String	
Length	8	Decimal Places	0

- Note that the descriptions given in the tab “Field Label” is how these elements will be labeled in the BPS Planning Parameter. Therefore, it is important to give both data elements meaningful descriptions.

Attributes	Data Type	Further Characteristics	Field Label
	Length	Field Label	
Short	10	Target Var	
Medium	19	BPS Target Variable	
Long	20	BPS Target Variable	
Heading	19	BPS Target Variable	

c) Create the source variable

Create a characteristic variable with restriction of value required by end. Assign this variable to profile(s) of users who authorized to select the global value.

Example:

Create variable	
Variable	ZADBVER
Description	Administrator Budget Version
Variable Type	Characteristic Value
<input checked="" type="checkbox"/> Create <input type="checkbox"/> Cancel	

Characteristics

Text Replacement

Replacement Type

Restriction of Values Required by User

Selection Conditions

	C..Version	To
<input type="checkbox"/>	= 1	
<input type="checkbox"/>	= 2	
<input type="checkbox"/>	= 3	
<input type="checkbox"/>	= 4	
<input type="checkbox"/>	= 5	

d) Create fixed value target variable

Create a fixed value variable, with the same characteristic(s) as the source variable, as the target.

Example:















Create variable

Variable

Description

Variable Type

Create





Characteristics	Version	 
Text Replacement	Medium text	
Replacement Type	Fixed Value	
<input type="checkbox"/> Restriction of Values Required by User		
         		
Selection Conditions		
	C..Version	To
	= 1	

e) Create a BPS customer exit planning function

i) Create a BPS planning function of type “Customer Exit”.

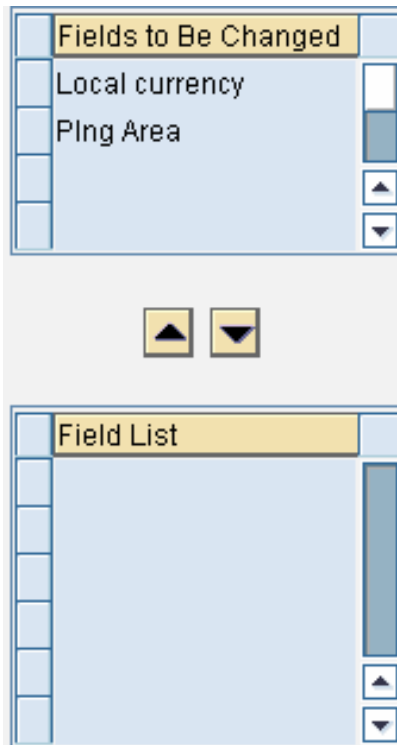
- *HINT: It is best to create this function in a level with a limited number of data records, since you will not actually be processing any data records, and the larger the number of data records, the longer the runtime for the function.*

ii) Specify the Function Module “Z_BPS_DERIVE_VAR”.

Planning function	ZVARDIR	DERIVE ONE VARIABLE VALUE FRO ...	 
Planning func. type	Exit Function		
Function module	Z_BPS_DERIVE_VAR		
FM initialization			

iii) Include all characteristics in the “Fields To Be Changed”.

- This ensures that the function will only run once, when called.



iv) Specify the following parameters:

- The planning function references these parameters by name in the constants declaration.

Parameter

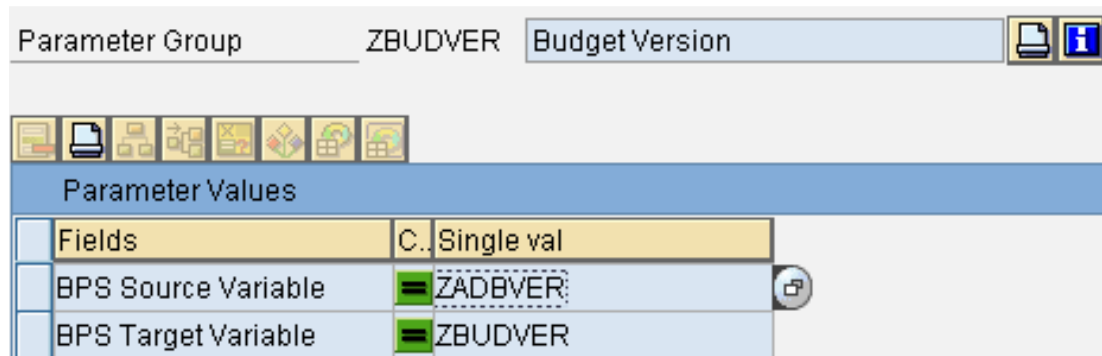
Name	Data Element
ZVARSRCE	Z_BPS_SRC_VAR
ZVARTAR	Z_BPS_TAR_VAR

Example:

Parameter exit functions	
Paramet...	Data element
ZVARSRCE	Z_BPS_SRC_VAR
ZVARTAR	Z_BPS_TAR_VAR

f) Create a separate parameter group for each variable you wish to derive

- i) Under the function, “Z_BPS_DERIVE_VAR”, create a parameter.
- ii) In the parameter group, specify the user specific variable you wish to derive from in the field “BPS Source Variable”
- iii) Specify the fixed value variable you wish to derive in the field “BPS Target Variable”.



g) Include this function in the profile for the administrative user.

HINT: The administrative user can execute this function directly from a push button on planning folder/web interface page. However, it may also be useful to include this function in a planning function executed on save. Therefore, when an administrator saves his data, the derived variable values are automatically applied to all users.

4) Appendix A: Function Module “Z_BPS_DERIVE_VAR”

```
FUNCTION Z_BPS_DERIVE_VAR.
```

```

*-----
*""Local Interface:
*  IMPORTING
*    REFERENCE(I_AREA) TYPE  UPC_Y_AREA
*    REFERENCE(I_PLEVEL) TYPE  UPC_Y_PLEVEL
*    REFERENCE(I_METHOD) TYPE  UPC_Y_METHOD
*    REFERENCE(I_PARAM) TYPE  UPC_Y_PARAM
*    REFERENCE(I_PACKAGE) TYPE  UPC_Y_PACKAGE
*    REFERENCE(IT_EXITP) TYPE  UPF_YT_EXITP
*    REFERENCE(ITO_CHASEL) TYPE  UPC_YTO_CHASEL
*    REFERENCE(ITO_CHA) TYPE  UPC_YTO_CHA
*    REFERENCE(ITO_KYF) TYPE  UPC_YTO_KYF
*  EXPORTING

```

```

*"      REFERENCE(ET_MESG) TYPE  UPC_YT_MESG
*"      CHANGING
*"      REFERENCE(XTH_DATA) TYPE  HASHED TABLE
*"-----
*The names of the BPS parameters
  CONSTANTS: VAR_SOURCE_PARAM  TYPE  UPC_Y_PARAM VALUE 'ZVARSRCE',
             VAR_TARGET_PARAM TYPE  UPC_Y_PARAM VALUE 'ZVARTAR'.

*Variables for function parameter processing
  DATA: LS_EXITP TYPE UPF_YS_EXITP,
         L_TARGET_VAR TYPE UPC_Y_VARIABLE,
         L_SOURCE_VAR TYPE UPC_Y_VARIABLE.

*Variables for source BPS variable processing
  DATA: ls_varsel TYPE upc_ys_charsel,
         LTO_VARSEL TYPE UPC_YTO_CHARSSEL,
         l_entries TYPE i,
         ls_mesg TYPE UPC_YS_MESG,
         l_subrc LIKE SY-SUBRC,
         L_Msg_string TYPE STRING.

*BPS Variable class variables, for target variable processing
  DATA: LI_VAR TYPE REF TO CL_SEM_VARIABLE,
         LTO_CHANM TYPE UPC_YTO_CHA,
         L_TYPE TYPE UPC_Y_VAR_TYPE,
         L_VAR_REPLACE TYPE UPC_Y_VAR_REPLACE_TYPE,
         L_USER_VALUES TYPE UPC_Y_USERSEL_ENABLED.

*** Derive one variable from another

* Get Variable source from function parameters
  READ TABLE IT_EXITP INTO LS_EXITP WITH KEY
    PARNM = VAR_SOURCE_PARAM.

*If such a param does not exist, error
  if SY-SUBRC <> 0.
    CLEAR ls_mesg.

    MESSAGE e001(upf) INTO L_Msg_string
      WITH 'Please include param'
        VAR_SOURCE_PARAM
        'in BPS func.'.

    MOVE-CORRESPONDING syst TO ls_mesg.
    APPEND ls_mesg TO et_mesg.
    RETURN.
  ENDIF.

*If param is not filled, error
  IF LS_EXITP-CHAVL NE '#'.
    L_SOURCE_VAR = LS_EXITP-CHAVL.
  ELSE.

    CLEAR ls_mesg.
    MESSAGE e001(upf) INTO L_Msg_string
      WITH 'Please fill param'
        VAR_SOURCE_PARAM
        'in BPS func.'.
    MOVE-CORRESPONDING syst TO ls_mesg.

```

```

APPEND ls_mesg TO et_mesg.
RETURN.
ENDIF.

```

*Get value of the source variable

```

CALL FUNCTION 'Z_VARIABLE_GET_DETAIL'
EXPORTING
  i_area      = i_area
  i_variable  = L_SOURCE_VAR
  i_buffer    = ''
IMPORTING
  e_subrc    = l_subrc
  eto_varse1 = LTO_VARSEL.

```

```
IF l_subrc <> 0.
```

* Values of variable &l cannot be determined

```
CLEAR ls_mesg.
```

```

MESSAGE e001(upf) INTO L_Msg_string
  WITH 'Variable'
  L_SOURCE_VAR
  'does not exists or contains no values.'.

```

```
MOVE-CORRESPONDING syst TO ls_mesg.
```

```
APPEND ls_mesg TO et_mesg.
```

```
RETURN.
```

```
ENDIF.
```

```
DESCRIBE TABLE lto_varse1 LINES l_entries.
```

```
IF l_entries = 0.
```

*Variable contains no values, send error message

```
CLEAR ls_mesg.
```

```
MESSAGE e001(upf) INTO L_Msg_string
```

```
  WITH 'Variable'
```

```
  L_SOURCE_VAR
```

```
  'contains no values.'.

```

```
MOVE-CORRESPONDING syst TO ls_mesg.
```

```
APPEND ls_mesg TO et_mesg.
```

```
RETURN.
```

```
ENDIF.
```

* Get Target Variable to set from function parameters

```
READ TABLE IT_EXITP INTO LS_EXITP WITH KEY
```

```
  PARNM = VAR_TARGET_PARAM .
```

*If such a param does not exist, error

```
if SY-SUBRC <> 0.
```

```
CLEAR ls_mesg.
```

```
MESSAGE e001(upf) INTO L_Msg_string
```

```
  WITH 'Please include param'
```

```
  VAR_TARGET_PARAM
```

```
  'in BPS func.'.

```

```
MOVE-CORRESPONDING syst TO ls_mesg.
```

```
APPEND ls_mesg TO et_mesg.
```

```
RETURN.
```

```
ENDIF.
```

*If param is not filled, error

```
IF LS_EXITP-CHAVL NE '#'.

```

```

L_TARGET_VAR = LS_EXITP-CHAVL.
ELSE.
CLEAR ls_mesg.
MESSAGE e001(upf) INTO L_Msg_string
  WITH 'Please fill param'
  VAR_TARGET_PARAM.

MOVE-CORRESPONDING syst TO ls_mesg.
APPEND ls_mesg TO et_mesg.
RETURN.
ENDIF.

```

***Get variable for update**

```

CALL METHOD CL_SEM_VARIABLE=>GET_INSTANCE
EXPORTING
  I_AREA      = I_AREA
  I_VARIABLE  = L_TARGET_VAR
RECEIVING
  RR_VARIABLE = LI_VAR
EXCEPTIONS
  NOT_EXISTING = 1
  others      = 2.
IF SY-SUBRC <> 0.
MOVE-CORRESPONDING SY TO LS_MESG.
APPEND LS_MESG TO ET_MESG.
RETURN.
ENDIF.

```

***Get Attributes to check that variable is correctly formed**

```

CALL METHOD LI_VAR->GET_ATTRIBUTES
IMPORTING
  E_TYPE = L_TYPE
  E_REPLACE = L_VAR_REPLACE
  E_USER_VALUES = L_USER_VALUES
  ETO_CHANM = LTO_CHANM.

```

***Check that the target variable is type fixed Char Value,
* with user input not allowed.**

```

IF L_TYPE NE 'CHAR' OR
L_VAR_REPLACE NE 'CHARVALUE' OR
L_USER_VALUES IS NOT INITIAL.

CLEAR ls_mesg.
MESSAGE e001(upf) INTO L_Msg_string
  WITH 'Variable'
  L_TARGET_VAR
  ' isnt type fixed char value.'.
MOVE-CORRESPONDING syst TO ls_mesg.
APPEND ls_mesg TO et_mesg.
RETURN.

```

ENDIF.

***Check that the variable contains the correct characteristic**

```

LOOP AT LTO_VARSEL INTO LS_VARSEL.
  READ TABLE LTO_CHANM
  WITH TABLE KEY CHANM = LS_VARSEL-CHANM
  TRANSPORTING NO FIELDS.
  IF SY-SUBRC > 2.

```

*** The target variable does not contain the correct characteristic**

```
CLEAR ls_mesg.  
MESSAGE e001(upf) INTO L_Msg_string  
  WITH 'Variable'  
    L_TARGET_VAR  
    'does not contain characteristic'  
    LS_Varsel-CHANM.  
MOVE-CORRESPONDING syst TO ls_mesg.  
APPEND ls_mesg TO et_mesg.  
RETURN.  
ENDIF.  
ENDLOOP.  
  
*Set the new values  
CALL METHOD LI_VAR->SET_ATTRIBUTES  
  EXPORTING  
    ITO_SEL = LTO_VARSEL.  
  
ENDFUNCTION.
```