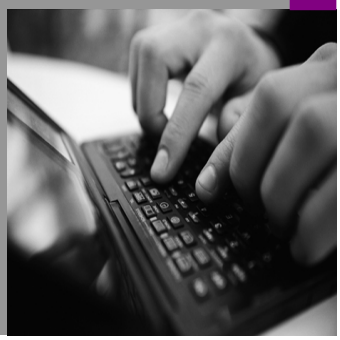


How-to Guide
SAP NetWeaver '04

How to... Upload User-Specific Variable Selections in BW-BPS

Version 1.00 – June 2004

Applicable Releases:
SAP NetWeaver '04
(SAP BW 3.5)

THE BEST-RUN BUSINESSES RUN SAP

© Copyright 2004 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, and Informix are trademarks or registered trademarks of IBM Corporation in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

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. SAP 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.
SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP 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.
SAP NetWeaver “How-To” Guides are intended to simplify the product implementation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using SAP NetWeaver. Should you wish to receive additional information, clarification or support, please refer to SAP Consulting.

1 Business Scenario

This document describes how to load a flat file containing the selection criteria for user-specific variables into BW-BPS. The provided program can be useful for the initial setup of user-specific variable values especially if the number of users is high. The program can also be used to maintain the variable values instead of changing them in the Planning Framework (for example if changes in the production system are not allowed due to authorization restrictions).

1. Preliminary remarks

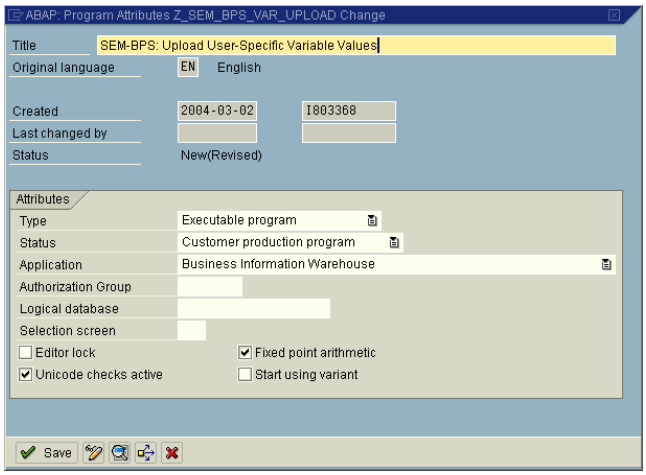
The solution requires you to implement an ABAP program. Therefore you need to have basic programming skills and the authorizations to do the programming in your system.

2. Create a new program

Run transaction SE38.

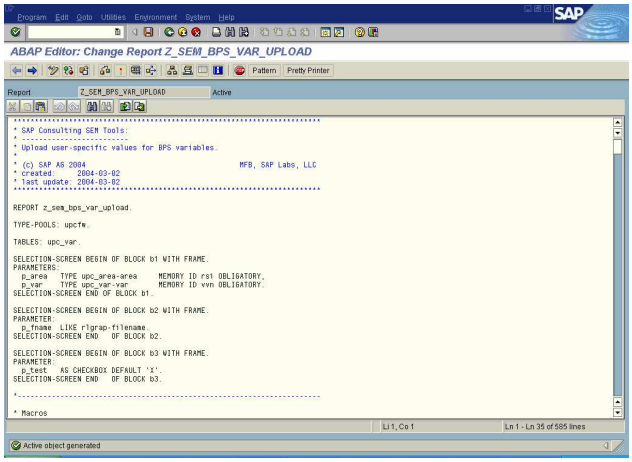
Enter program name Z_BPS_VAR_UPLOAD and click “create”. Enter title and type and save it.

Select a package and save the definition to a transport.



3. Activate the program

Copy the source code from appendix A into the editor and activate the program.

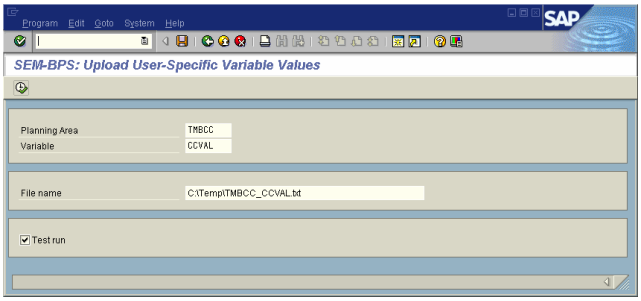


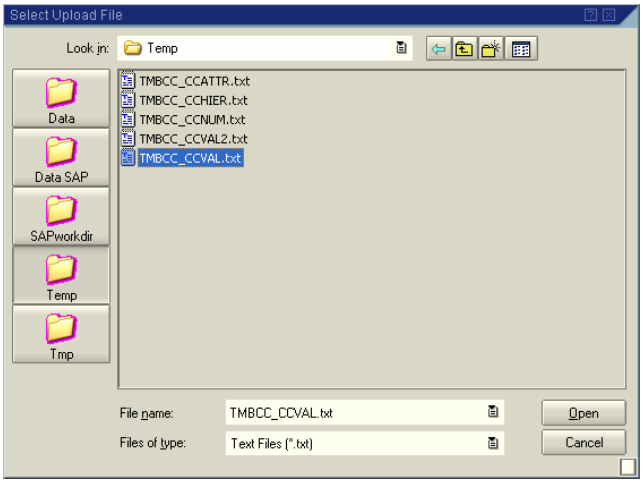
2 The Step By Step Solution

1. Start the program

Enter the planning area and variable that you want to populate from the flat file.

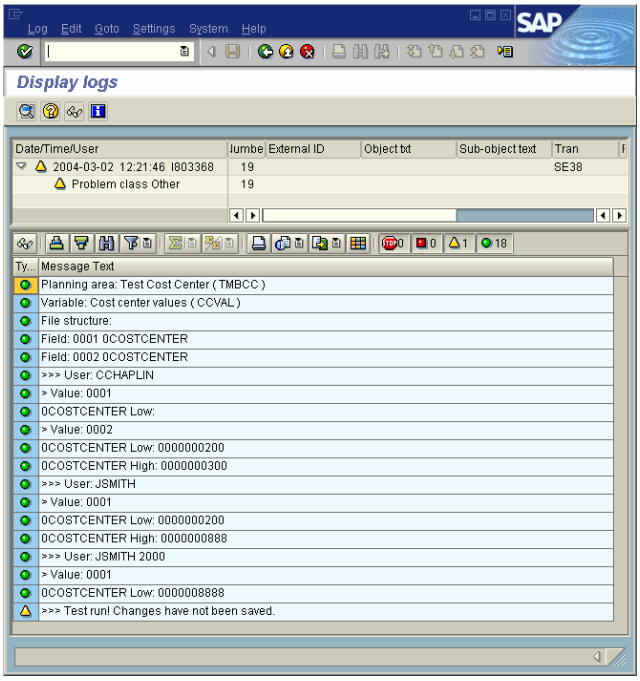
The file can be selected using the F4 value-help.





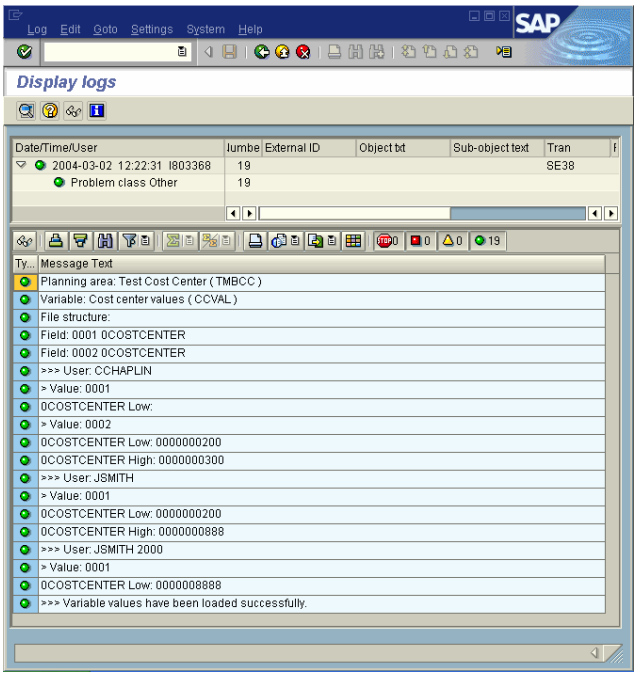
2. Execute the upload in test mode

It is recommended to run the upload in test mode the first time. This will load the file and validate the content but will not save the selections to the database.



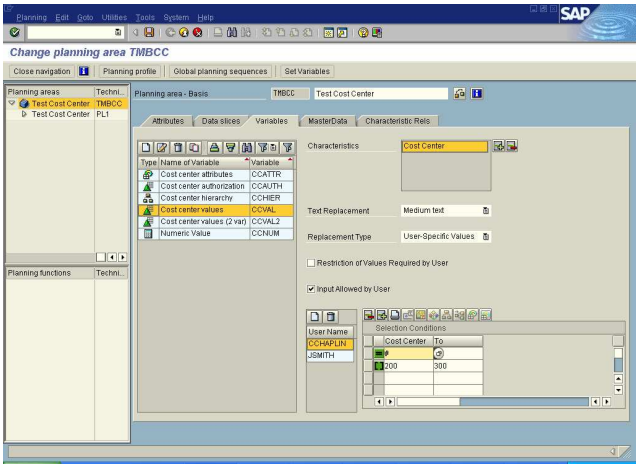
3. Execute the upload

If now error messages were raised during test mode, turn the test mode off and load the file again.



4. Check if the upload was successful

In the Planning Framework (transaction BPS0) you can view the selection criteria that have been loaded for each user ID.



3 Comments

- The upload will replace existing selection criteria for the user ids that are given in the file.
- If the user had restricted the selection to a single value (tuple), then the restriction is removed and the user has to restrict the selection again. The reason is that the restricted value might not exist in the newly loaded selections.

4 Appendix A

```
************************************************************************
* Upload user-specific values for BPS variables.
*
* (c) SAP AG 2004
* last update: 2004-06-02
************************************************************************
* Text elements:
* P_AREA   Planning Area
* P_FNAME  File name
* P_TEST   Test run
* P_VAR    Variable
************************************************************************

REPORT z_bps_var_upload.

TYPE-POOLS: upcfw.

TABLES: upc_var.

SELECTION-SCREEN BEGIN OF BLOCK b1 WITH FRAME.
PARAMETERS:
  p_area   TYPE upc_area-area     MEMORY ID rsi OBLIGATORY,
  p_var    TYPE upc_var-var       MEMORY ID vvn OBLIGATORY.
SELECTION-SCREEN END OF BLOCK b1.

SELECTION-SCREEN BEGIN OF BLOCK b2 WITH FRAME.
PARAMETER:
  p_fname  LIKE rlgrap-filename.
SELECTION-SCREEN END   OF BLOCK b2.

SELECTION-SCREEN BEGIN OF BLOCK b3 WITH FRAME.
PARAMETER:
  p_test   AS CHECKBOX DEFAULT 'X'.
SELECTION-SCREEN END   OF BLOCK b3.

*---------------------------------------------------------------------

* Macros
DATA:
  g_dummy(72)   TYPE c,
  g_debug(1)    TYPE c,
  g_subrc       LIKE sy-subrc,
  gs_bal_msg    TYPE bal_s_msg,
  gs_bal_log    TYPE bal_s_log,
  g_loghandle   TYPE balloghndl,
  gt_loghandle  TYPE bal_t_logh.

* Create new application log
DEFINE log_open.
  call function 'BAL_LOG_CREATE'
    exporting
      i_s_log      = gs_bal_log
    importing
      e_log_handle = g_loghandle.

  append g_loghandle to gt_loghandle.
END-OF-DEFINITION.

* Display application log
DEFINE log_close.
  call function 'BAL_DSP_LOG_DISPLAY'
    exporting
```

```
      i_t_log_handle = gt_loghandle.
END-OF-DEFINITION.

* Add message to application log
* When in batch mode, add message to job log as well
DEFINE log_add.
  g_subrc = sy-subrc.

  if sy-batch = 'X'.
*   Always use "I"-messages (all others cause jobs to be cancelled)
    message id sy-msgid type 'I' number sy-msgno
            with sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  endif.
  move-corresponding syst to gs_bal_msg.

  if not gs_bal_msg is initial.
    call function 'BAL_LOG_MSG_ADD'
      exporting
        i_log_handle = g_loghandle
        i_s_msg      = gs_bal_msg.
  endif.

  clear: sy-msgid, sy-msgno, sy-msgty, sy-subrc,
         sy-msgv1, sy-msgv2, sy-msgv3, sy-msgv4.
  clear: gs_bal_msg.
END-OF-DEFINITION.

* Add message and exit
DEFINE log_add_exit.
  log_add.
  exit.
END-OF-DEFINITION.

* Add message and exit in case of error
DEFINE log_add_exit_subrc.
  if sy-subrc <> 0.
    log_add.
    exit.
  endif.
END-OF-DEFINITION.

* Append generic message
DEFINE log_add_msg.
  if &1 = 'I'.
    message i001(upf) into g_dummy with &2 &3 &4.
    sy-subrc = 0.
  elseif &1 = 'W'.
    message w001(upf) into g_dummy with &2 &3 &4.
    sy-subrc = 2.
  elseif &1 = 'E'.
    message e001(upf) into g_dummy with &2 &3 &4.
    sy-subrc = 4.
  endif.
  log_add.
END-OF-DEFINITION.

*---------------------------------------------------------------------

* File
TYPES:
  BEGIN OF ys_file,
    varuser TYPE uname,
    value01 TYPE upc_y_chavlext,
    value02 TYPE upc_y_chavlext,
    value03 TYPE upc_y_chavlext,
    value04 TYPE upc_y_chavlext,
    value05 TYPE upc_y_chavlext,
```

```
    value06 TYPE upc_y_chavlext,
    value07 TYPE upc_y_chavlext,
    value08 TYPE upc_y_chavlext,
    value09 TYPE upc_y_chavlext,
    value10 TYPE upc_y_chavlext,
    value11 TYPE upc_y_chavlext,
    value12 TYPE upc_y_chavlext,
    value13 TYPE upc_y_chavlext,
    value14 TYPE upc_y_chavlext,
    value15 TYPE upc_y_chavlext,
    value16 TYPE upc_y_chavlext,
    value17 TYPE upc_y_chavlext,
    value18 TYPE upc_y_chavlext,
    value19 TYPE upc_y_chavlext,
    value20 TYPE upc_y_chavlext,
  END OF ys_file,

  BEGIN OF ys_field,
    seqno TYPE upc_y_seqno,
    chanm TYPE upc_y_chanm,
  END OF ys_field.

DATA:
  g_fname             TYPE string,
  gs_file             TYPE ys_file,
  gt_file             TYPE STANDARD TABLE OF ys_file,
  gs_field            TYPE ys_field,
  gt_field            TYPE STANDARD TABLE OF ys_field.

* Variable definition
DATA:
  gr_var              TYPE REF TO cl_sem_variable,
  g_user              TYPE sy-uname,
  g_area_text         TYPE upc_y_areatxt,
  g_var_text          TYPE upc_y_variabletxt,
  g_type              TYPE upc_y_var_type,
  g_replace           TYPE upc_y_var_replace_type,
  gs_chanm            TYPE upc_ys_cha,
  gto_chanm           TYPE upc_yto_cha,
  gto_attr            TYPE upc_yto_attr,
  gs_hierarchy        TYPE upc_ys_hie_key.

* Variable values
DATA:
  g_seqno             TYPE upc_y_seqno,
  g_chanm             TYPE upc_y_chanm,
  g_count             TYPE sy-dbcnt,
  g_tabix             TYPE sy-tabix,
  g_lowhigh           TYPE sy-tabix,
  g_field             TYPE upc_y_chavlext,
  g_chavlext          TYPE upc_y_chavlext,
  g_chavlint          TYPE upc_y_chavlint,
  gs_charsel          TYPE upc_ys_usercharsel,
  gto_charsel         TYPE upc_yto_usercharsel,
  gto_charsel_act     TYPE upc_yto_usercharsel,
  gs_hiesel           TYPE upc_ys_userhiesel,
  gto_hiesel          TYPE upc_yto_userhiesel,
  gto_hiesel_act      TYPE upc_yto_userhiesel,
  gs_chadep           TYPE upc_ys_chadep,
  gto_chadep          TYPE upc_yto_chadep,
  gs_numsel           TYPE upc_ys_usernumsel,
  gto_numsel          TYPE upc_yto_usernumsel.

FIELD-SYMBOLS:
  <g_field>           TYPE ANY.

*---------------------------------------------------------------------
```

```
AT SELECTION-SCREEN ON VALUE-REQUEST FOR p_fname.

  DATA:
    lt_filetab  TYPE filetable,
    ls_filetab  TYPE file_table,
    l_action    TYPE i,
    l_count     TYPE i.

  CALL METHOD cl_gui_frontend_services=>file_open_dialog
    EXPORTING
      window_title            = 'Select Upload File'
      default_extension       = 'txt'
      file_filter             = 'Text Files (*.txt)'
    CHANGING
      file_table              = lt_filetab
      rc                      = l_count
      user_action             = l_action
    EXCEPTIONS
      file_open_dialog_failed = 1
      cntl_error              = 2
      OTHERS                  = 3.                          "#EC NOTEXT
  CHECK sy-subrc = 0 AND l_action = 0.

  CALL METHOD cl_gui_cfw=>flush.

  LOOP AT lt_filetab INTO ls_filetab.
    p_fname = ls_filetab.
  ENDLOOP.

*---------------------------------------------------------------------
START-OF-SELECTION.

  log_open.

* Get planning area
  CALL FUNCTION 'UPC_AREA_GET'
    EXPORTING
      i_area       = p_area
    IMPORTING
      e_areat      = g_area_text
    EXCEPTIONS
      no_existence = 1
      locked       = 2
      OTHERS       = 3.
  log_add_exit_subrc.

* Get variable definition
  CALL METHOD cl_sem_variable=>get_instance
    EXPORTING
      i_area       = p_area
      i_variable   = p_var
    RECEIVING
      rr_variable  = gr_var
    EXCEPTIONS
      not_existing = 1
      OTHERS       = 2.
  log_add_exit_subrc.

  CALL METHOD gr_var->get_attributes
    IMPORTING
      e_text       = g_var_text
      e_type       = g_type
      e_replace    = g_replace
      eto_chanm    = gto_chanm
      eto_attr     = gto_attr
      es_hierarchy = gs_hierarchy
      eto_usersel  = gto_charsel
```

```
      eto_hiesel   = gto_hiesel
      eto_numsel   = gto_numsel.

  CASE g_type.
    WHEN upcfw_cs_var_type-char.
      DESCRIBE TABLE gto_chanm LINES g_count.

    WHEN upcfw_cs_var_type-attribute.
      DESCRIBE TABLE gto_attr LINES g_count.

      READ TABLE gto_chanm INTO gs_chanm INDEX 1.
      IF sy-subrc = 0.
        g_chanm = gs_chanm-chanm.
      ENDIF.
  ENDCASE.

* Messages
  CONCATENATE '(' p_area ')' INTO g_dummy SEPARATED BY space.
  log_add_msg 'I' 'Planning area:' g_area_text g_dummy.

  CONCATENATE '(' p_var ')' INTO g_dummy SEPARATED BY space.
  log_add_msg 'I' 'Variable:' g_var_text g_dummy.

* Check variable
  IF g_replace <> upcfw_cs_var_replace-uservalue.
    log_add_msg 'E' 'Only variables with replacement type'
                    '"User-specific value" can be loaded.' ''.
    EXIT.
  ENDIF.

* Read file
  g_fname = p_fname.

  CALL FUNCTION 'GUI_UPLOAD'
    EXPORTING
      filename                = g_fname
      filetype                = 'ASC'
      has_field_separator     = 'X'
    TABLES
      data_tab                = gt_file
    EXCEPTIONS
      file_open_error         = 1
      file_read_error         = 2
      no_batch                = 3
      gui_refuse_filetransfer = 4
      invalid_type            = 5
      no_authority            = 6
      unknown_error           = 7
      bad_data_format         = 8
      header_not_allowed      = 9
      separator_not_allowed   = 10
      header_too_long         = 11
      unknown_dp_error        = 12
      access_denied           = 13
      dp_out_of_memory        = 14
      disk_full               = 15
      dp_timeout              = 16
      OTHERS                  = 17.
  log_add_exit_subrc.

* Get column headings from first row (characteristics/attributes)
  READ TABLE gt_file INTO gs_file INDEX 1.
  IF sy-subrc <> 0.
    log_add_msg 'E' 'File is empty' '' ''.
    EXIT.
  ENDIF.
```

```
  log_add_msg 'I' 'File structure:' '' ''.

  CLEAR g_seqno.
  REFRESH gt_field.
  DO 20 TIMES VARYING g_field FROM gs_file-value01 NEXT gs_file-value02.
    CHECK NOT g_field IS INITIAL.

    ADD 1 TO g_seqno.

    CLEAR gs_field.
    gs_field-seqno = g_seqno.
    gs_field-chanm = g_field.
    APPEND gs_field TO gt_field.

    log_add_msg 'I' 'Field:' g_seqno g_field.
  ENDDO.

  DELETE gt_file INDEX 1.

* Now sort by user
  SORT gt_file BY varuser.

* Convert file structure in variable selections
  LOOP AT gt_file INTO gs_file.
*   Next user
    AT NEW varuser.
      g_seqno = 0.
      g_user  = gs_file-varuser.
      CHECK NOT g_user IS INITIAL.

      log_add_msg 'I' '>>> User:' g_user ''.

      DELETE gto_charsel WHERE user = g_user.
      IF sy-subrc = 0.
        log_add_msg 'W' 'Existing selections for char. values deleted!' '' ''.
      ENDIF.

      DELETE gto_hiesel WHERE user = g_user.
      IF sy-subrc = 0.
        log_add_msg 'W' 'Existing selections for hier. nodes deleted!' '' ''.
      ENDIF.

      DELETE gto_numsel WHERE user = g_user.
      IF sy-subrc = 0.
        log_add_msg 'W' 'Existing selections for numeric values deleted!' '' ''.
      ENDIF.

    ENDAT.

*   Next value (tuple)
    ADD 1 TO g_seqno.

    log_add_msg 'I' '> Value:' g_seqno ''.

*   Compile user-specific values
    CASE g_type.

      WHEN upcfw_cs_var_type-char OR upcfw_cs_var_type-attribute.

        LOOP AT gt_field INTO gs_field.
          g_tabix = sy-tabix + 1.
          ASSIGN COMPONENT g_tabix OF STRUCTURE gs_file TO <g_field>.
          CHECK sy-subrc = 0.

          g_chavlext = <g_field>.
          IF g_chavlext IS INITIAL.
            CONTINUE.
```

```
      ELSEIF g_chavlext = '#'.
        CLEAR g_chavlext.
      ENDIF.

      IF g_type = upcfw_cs_var_type-char.
        CALL FUNCTION 'UPC_CHAVL_EX_IN_CONVERT'
          EXPORTING
            i_area          = p_area
            i_chanm         = gs_field-chanm
            i_chavlext      = g_chavlext
          IMPORTING
            e_chavlint      = g_chavlint
          EXCEPTIONS
            invalid_format  = 1
            chanm_not_found = 2
            rfc_error       = 3
            failed          = 4
            OTHERS          = 5.
        log_add_exit_subrc.
      ELSE.
        CALL FUNCTION 'UPC_ATRVL_EX_IN_CONVERT'
          EXPORTING
            i_area          = p_area
            i_chanm         = g_chanm
            i_atrnm         = gs_field-chanm
            i_atrvlext      = g_chavlext
          IMPORTING
            e_atrvlint      = g_chavlint
          EXCEPTIONS
            invalid_format  = 1
            atrnm_not_found = 2
            rfc_error       = 3
            failed          = 4
            OTHERS          = 5.
        log_add_exit_subrc.
      ENDIF.

      g_lowhigh = g_tabix MOD 2.
      IF g_lowhigh = 0.
*       Low value
        CLEAR gs_charsel.
        gs_charsel-user  = g_user.
        gs_charsel-chanm = gs_field-chanm.
        gs_charsel-seqno = g_seqno.
        gs_charsel-sign  = 'I'.
        gs_charsel-opt   = 'EQ'.
        gs_charsel-low   = g_chavlint.
        INSERT gs_charsel INTO TABLE gto_charsel.

        log_add_msg 'I' gs_field-chanm 'Low:' g_chavlint.
      ELSE.
*       High value
        READ TABLE gto_charsel INTO gs_charsel WITH TABLE KEY
          user  = g_user
          chanm = gs_field-chanm
          seqno = g_seqno.
        IF sy-subrc = 0 AND g_chavlint >= gs_charsel-low.
          gs_charsel-opt  = 'BT'.
          gs_charsel-high = g_chavlint.
          MODIFY gto_charsel FROM gs_charsel INDEX sy-tabix.

          log_add_msg 'I' gs_field-chanm 'High:' g_chavlint.
        ELSE.
          log_add_msg 'E' 'Incorrect "HIGH" value:' g_chavlint ''.
        ENDIF.
      ENDIF.
    ENDLOOP.
```

```
WHEN upcfw_cs_var_type-hierarchy.

  REFRESH gto_chadep.
  LOOP AT gt_field INTO gs_field FROM 4.
    g_tabix = sy-tabix + 1.
    ASSIGN COMPONENT g_tabix OF STRUCTURE gs_file TO <g_field>.
    CHECK sy-subrc = 0.

    g_chavlext = <g_field>.
    IF g_chavlext IS INITIAL.
      CONTINUE.
    ELSEIF g_chavlext = '#'.
      CLEAR g_chavlext.
    ENDIF.

    CALL FUNCTION 'UPC_CHAVL_EX_IN_CONVERT'
      EXPORTING
        i_area          = p_area
        i_chanm         = gs_field-chanm
        i_chavlext      = g_chavlext
      IMPORTING
        e_chavlint      = g_chavlint
      EXCEPTIONS
        invalid_format  = 1
        chanm_not_found = 2
        rfc_error       = 3
        failed          = 4
        OTHERS          = 5.
    log_add_exit_subrc.

    CLEAR gs_chadep.
    gs_chadep-chanm    = gs_field-chanm.
    gs_chadep-chavlint = g_chavlint.
    INSERT gs_chadep INTO TABLE gto_chadep.
  ENDLOOP.

  IF gs_file-value01 = gs_hierarchy-chanm.
    g_chavlext = gs_file-value02.
    CALL FUNCTION 'UPC_CHAVL_EX_IN_CONVERT'
      EXPORTING
        i_area          = p_area
        i_chanm         = gs_hierarchy-chanm
        i_chavlext      = g_chavlext
      IMPORTING
        e_chavlint      = g_chavlint
      EXCEPTIONS
        invalid_format  = 1
        chanm_not_found = 2
        rfc_error       = 3
        failed          = 4
        OTHERS          = 5.
    log_add_exit_subrc.
  ELSE.
    g_chavlint = gs_file-value02.
  ENDIF.

  CLEAR gs_hiesel.
  MOVE-CORRESPONDING gs_hierarchy TO gs_hiesel.
  gs_hiesel-user       = g_user.
  gs_hiesel-seqno      = g_seqno.
  gs_hiesel-hiecha     = gs_file-value01.
  gs_hiesel-nodename   = g_chavlint.
  gs_hiesel-dummy_leaf = gs_file-value03.
  gs_hiesel-to_chadep  = gto_chadep.
  INSERT gs_hiesel INTO TABLE gto_hiesel.
```

```
        log_add_msg 'I' 'Node:' gs_hiesel-hiecha gs_hiesel-nodename.

      WHEN upcfw_cs_var_type-number.

        CLEAR gs_numsel.
        gs_numsel-user  = g_user.
        gs_numsel-value = gs_file-value01.
        IF gs_numsel-value IS INITIAL.
          gs_numsel-no_value = 'X'.
        ENDIF.
        INSERT gs_numsel INTO TABLE gto_numsel.

        log_add_msg 'I' 'Number:' gs_numsel-value ''.

    ENDCASE.

  ENDLOOP.

* Set values for variable
  CASE g_type.

    WHEN upcfw_cs_var_type-char OR upcfw_cs_var_type-attribute.

      CALL METHOD gr_var->set_attributes
        EXPORTING
          ito_usersel     = gto_charsel
          ito_usersel_act = gto_charsel_act.

    WHEN upcfw_cs_var_type-hierarchy.

      CALL METHOD gr_var->set_attributes
        EXPORTING
          ito_hiesel     = gto_hiesel
          ito_hiesel_act = gto_hiesel_act.

    WHEN upcfw_cs_var_type-number.

      CALL METHOD gr_var->set_attributes
        EXPORTING
          ito_numsel = gto_numsel.

  ENDCASE.

* Commit changes
  IF g_subrc = 0.
    IF p_test IS INITIAL.
      CALL METHOD cl_sem_variable=>save_db_all.
      log_add_msg 'I' '>>> Variable values have been loaded successfully.' '' ''
.
    ELSE.
      log_add_msg 'W' '>>> Test run!' 'Changes have not been saved.' ''.
    ENDIF.
  ENDIF.

*---------------------------------------------------------------------
END-OF-SELECTION.

  log_close.
```

5 Appendix B: File Formats and Examples

In general, the file has to be in **text-format** with **TAB-delimited** fields. The first row in the file has to contain the **fieldnames** according to the specification given below. The first column in each data row contains the **user ID**. All following columns determine the **user-specific selections** in **external format**.

5.1 Characteristic Value Variables

5.1.1 Header Row

Field Number	Field Value	Description
1	USER	Fixed value "USER"
2	<CHARACTERISTIC_1>	Technical name of the first characteristic
3	<CHARACTERISTIC_1>	Technical name of the first characteristic
4	<CHARACTERISTIC_2>	Technical name of the second characteristic
5	<CHARACTERISTIC_2>	Technical name of the second characteristic
…	…	…

Note: The order of the characteristics is not important. For each characteristic two columns have to be provided.

5.1.2 Data Row

Field Number	Field Value	Description
1	<USER_ID>	User ID
2	<VALUE_1_LOW>	Low value of the first characteristic
3	<VALUE_1_HIGH>	High value of the first characteristic
4	<VALUE_2_LOW>	Low value of the second characteristic
5	<VALUE_2_HIGH>	High value of the second characteristic
…	…	…

Note: If low and high values are provided, the high value must be greater than the low value. To specify the selection of "**not assigned**", set the field value to "**#**'.

5.1.3 Example 1: One Characteristic

USER	0COSTCENTER	0COSTCENTER
JSMITH	200	888
JSMITH	2000	8888
CCHAPLIN	#	
CCHAPLIN	200	300

5.1.4 Example 2: Several Characteristics

USER	0COSTCENTER	0COSTCENTER	0COSTELMNT	0COSTELMNT	0CO_AREA	0CO_AREA
JSMITH	200		400000		1000	
JSMITH	2000		400001		1000	
CCHAPLIN	100		500000		2000	
CCHAPLIN	200		500100		2000	

Note: If several characteristics are used, BPS will use **only single values** for each characteristic. I.e. ranges are not allowed although respective fields are provided in the file format.

5.2 Attribute Variables

5.2.1 Header Row

Field Number	Field Value	Description
1	USER	Fixed value "USER"
2	<ATTRIBUTE_1>	Technical name of the first attribute
3	<ATTRIBUTE_1>	Technical name of the first attribute
4	<ATTRIBUTE_2>	Technical name of the second attribute
5	<ATTRIBUTE_2>	Technical name of the second attribute
…	…	…

Note: The order of the attributes is not important. For each attribute two columns have to be provided.

5.2.2 Data Row

Field Number	Field Value	Description
1	<USER_ID>	User ID
2	<VALUE_1_LOW>	Low value of the first attribute
3	<VALUE_1_HIGH>	High value of the first attribute
4	<VALUE_2_LOW>	Low value of the second attribute
5	<VALUE_2_HIGH>	High value of the second attribute
…	…	…

Note: If low and high values are provided, the high value must be greater than the low value. To specify the selection of "**not assigned**", set the field value to "**#**'.

5.2.3 Example: Attributes of Cost Center

USER	0BUS_AREA	0BUS_AREA	0COMP_CODE	0COMP_CODE	0RESP_PERS	0RESP_PERS
JSMITH	1000		2400	2600	JSMITH	
JSMITH	2000					
CCHAPLIN	1000	2000	3000		CCHAPLIN	

5.3 Hierarchy Node Variables

5.3.1 Header Row

Field Number	Field Value	Description
1	USER	Fixed value "USER"
2	HIECHA	Fixed value "HIECHA"
3	NODENAME	Fixed value "NODENAME"
4	DUMMY_LEAD	Fixed value "DUMMY_LEAF"
5	<COMPOUND_CHAR_1>	Technical name of first compounded characteristic
6	<COMPOUND_CHAR_2>	Technical name of second compounded characteristic
…	…	…

Note: Compounded characteristics have to be specified only if the base characteristic of the hierarchy is compounded. The order of the compounded characteristics is not important.

5.3.2 Data Row

Field Number	Field Value	Description
1	<USER_ID>	User ID
2	<NODE_CHARACTERISTIC>	Technical name of the node characteristic
3	<NODE_VALUE>	Value of the node in the hierarchy
4	<FLAG_LEAF>	If the value is a leaf in the hierarchy, the flag must be set to X.
5	<COMPOUND_VALUE_1>	Value of first compounded characteristic
6	<COMPOUND_VALUE_2>	Value of second compounded characteristic
…	…	…

Note: To specify the selection of "**not assigned**", set the field value to "**#**". If the node is a **text node** (unpostable), then the node characteristic has to be set to "**0HIER_NODE**".

5.3.3 Example: Cost Center Hierarchy

USER	HIECHA	NODENAME	DUMMY_LEAF	0CO_AREA
JSMITH	0COSTCENTER	1000	X	1000
JSMITH	0COSTCENTER	2100		1000
JSMITH	0HIER_NODE	ROOT		
CCHAPLIN	0HIER_NODE	ROOT		

Note: Node 1000 is a single cost center as indicated by the leaf flag; node 2100 is a cost center group.

5.4 Numeric Value Variables

5.4.1 Header Row

Field Number	Field Value	Description
1	USER	Fixed value “USER”
2	NUMBER	Fixed value “NUMBER”

5.4.2 Data Row

Field Number	Field Value	Description
1	<USER_ID>	User ID
2	<NUMERIC_VALUE>	Numeric value

Note: To specify the selection of “**no number**”, leave the field value blank.

5.4.3 Example

USER	NUMBER
JSMITH	12345.678
CCHAPLIN	10000.001

www.sap.com/netweaver