# How to Enhance LISTCUBE Functionality

## Applies to

SAP BI 7.0 and SAP BW 3.x (with minor modifications)

## Summary

This document provides the detailed steps to enhance the functionality of the LISTCUBE transaction to speed up data analysis and troubleshooting involving Infoproviders with complex structures. The tool introduced in this document will be useful for System Analysts, Technical team and Key users who are involved in data analysis, data validation and troubleshooting.

**Author:** Manoj Vasudevan

**Company:** XB5 Consulting

**Created on:** 20 October 2007

## Author Bio

Manoj is a Senior SAP BI Consultant with more than 10 years of consulting experience in SAP modules. He has been involved in the projects of major MNC clients in the role of SAP BI Architect, Project Manager and Technology specialist covering FMCG, Oil and Gas, Manufacturing, Public Sector, Utilities and Electronics industries in Asia, US and Europe. Manoj has held senior positions in major consulting companies including PricewaterhouseCoopers where he was a Principal Consultant. He is currently a Director for Business Intelligence at XB5 Consulting. He helps clients in their SEM/BW implementations and provides advice on implementation, upgrade and maintenance strategies.

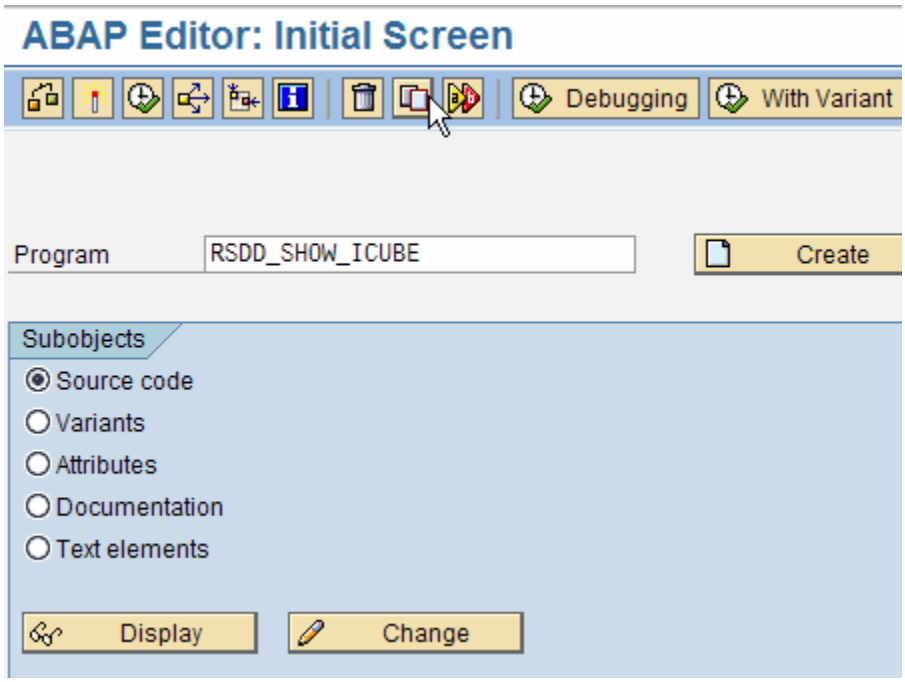
## Table of Contents

## Overview of the Tool

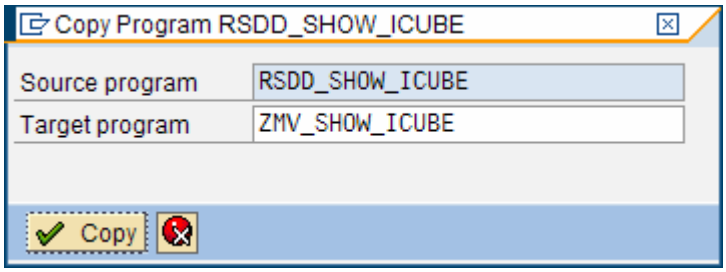
The LISTCUBE transaction in SAP-BI allows you to view the contents of Infoproviders. LISTCUBE is typically used for testing, troubleshooting and other data evaluation purposes. LISTCUBE can also be used to manually export Infoprovider content. However, when the data structures are complex, with many Characteristics, Navigational attributes and Keyfigures, the standard LISTCUBE functionality becomes bit cumbersome to use. For example, if an Infoprovider has more than 72 Characteristics and Navigational Attributes, the LISTCUBE transaction forces the user to restrict the selection to 72 Objects. This is fine for most cases. But if you have been involved in troubleshooting sessions for Infoproviders containing more than 72 Characteristics and Navigational Attributes, you would also know that it is difficult to select the required characteristics from a long list. Moreover, the selections you make cannot be saved for future use without generating a program (BI 7.0), which most Analysts would not have access to generate or execute in most Production environments. The following detailed steps show how to create a customized LISTCUBE transaction that contains added features that would be helpful to reduce turnaround times for testing and troubleshooting.

## Detailed Steps

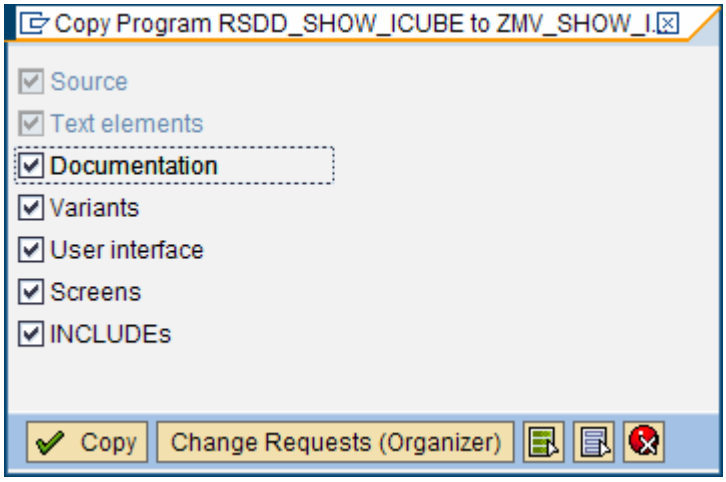
- Copy and enhance the SAP standard Program RSDD_SHOW_ICUBE to ZMV_SHOW_ICUBE.
    a. Goto Transaction SE38. Type the Program name RSDD_SHOW_ICUBE and select the Copy icon.



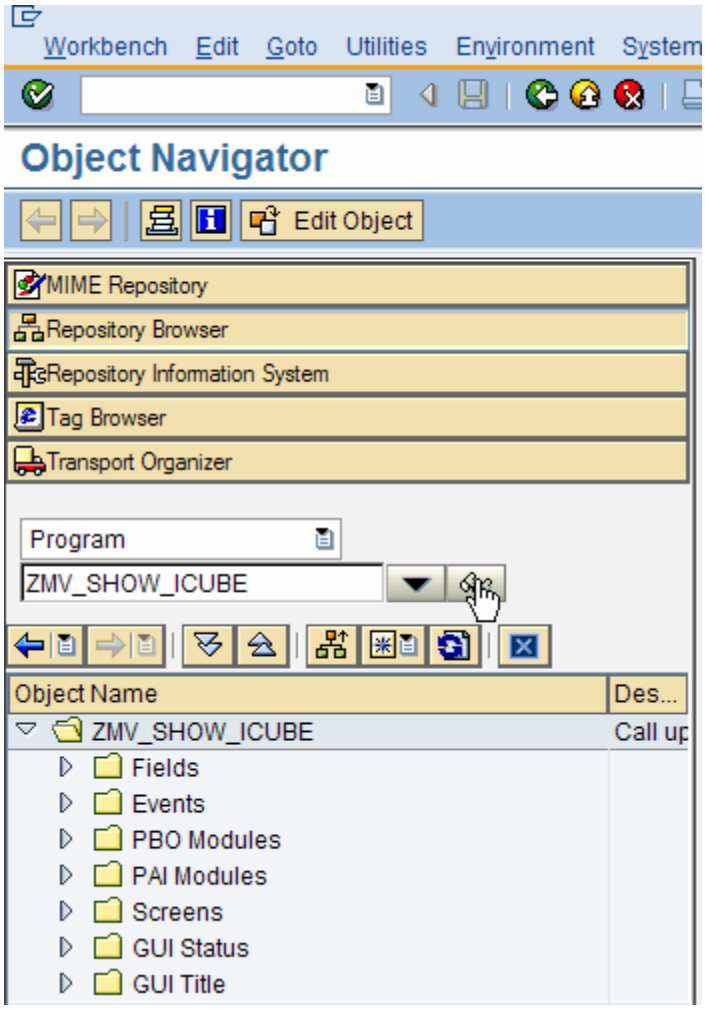

    b. Enter Target Program Name ZMV_SHOW_ICUBE



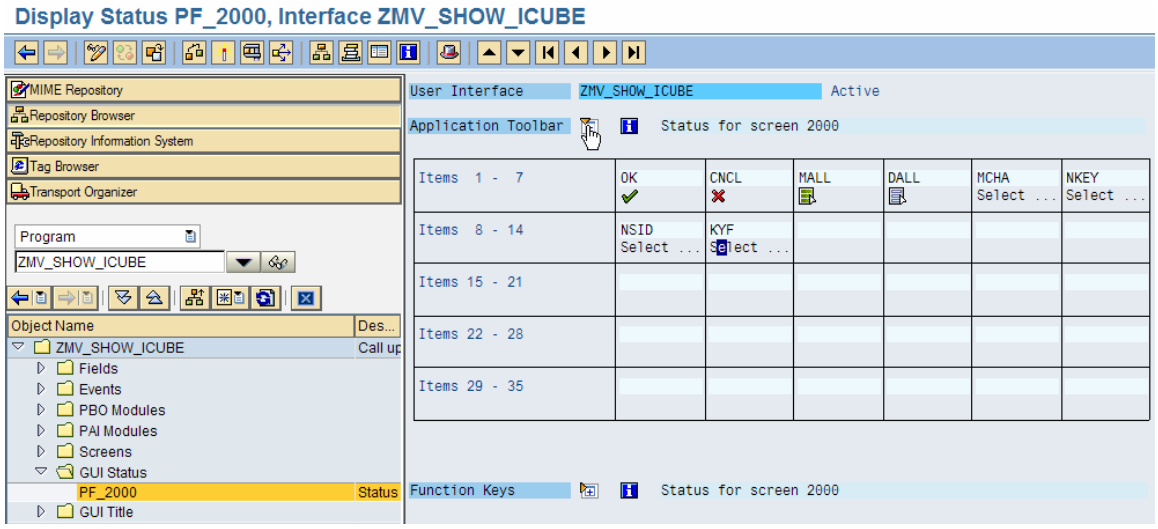

    c. Following Popup appears. Select All Checkboxes and Click *Copy*.



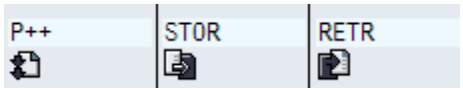

    d. Change the code of the program to the code given in **Appendix B** and Click *Save.*
    e. Go to transaction SE80. Select Program ZMV_SHOW_ICUBE for Display.



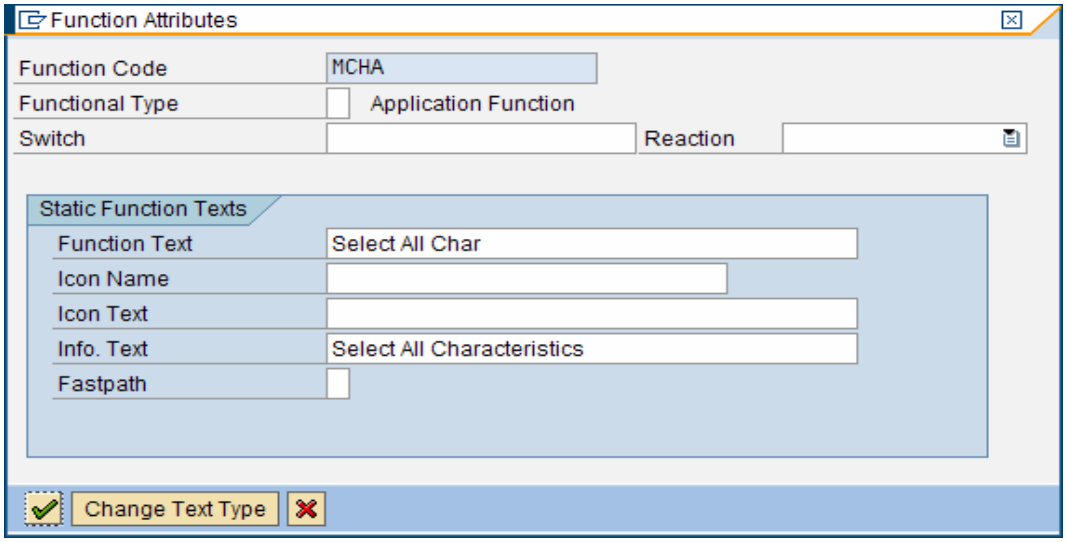

f. Now Select *GUI Status PF_2000* for amendment.




g. We now need to add the following Function Code and Icons in the Application Toolbar. These will appear in the *Characteristic selection popup* screen of this tool.




h. Since we are adding new Pushbuttons, the Toolbar will be cluttered unless we reduce the size of the *Function Texts* of the existing Pushbuttons. You can change the *Function Texts* by double-clicking on the *Function Codes*. Example for Function Code MCHA shown below. Here the *Function Text* is changed from "Select All Characteristics" to "Select All Char". This will reduce the length of the pushbutton in the Screen at runtime.




Recommendations for *Function Texts* for each *Function Code* are given below.

MCHA → Change "Select All Characteristics" to "Select All Char"

NKEY → Change "Select All Navigation Attributes" to "Select Nav Attr"

NSID → Change "Select the SIDs for All Navigation Attributes" to "Select Nav Attr SIDs"

KYF → Change "Select All Key Figures" to "Select Key Figs"

FIND → "Find Object"

FINN → "Find Next Object"

P-- → "First Page"

P- → "Previous Page"

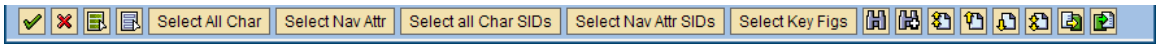
P+ → "Next Page"

P++ → "Last Page"

STOR → "Store Selections"

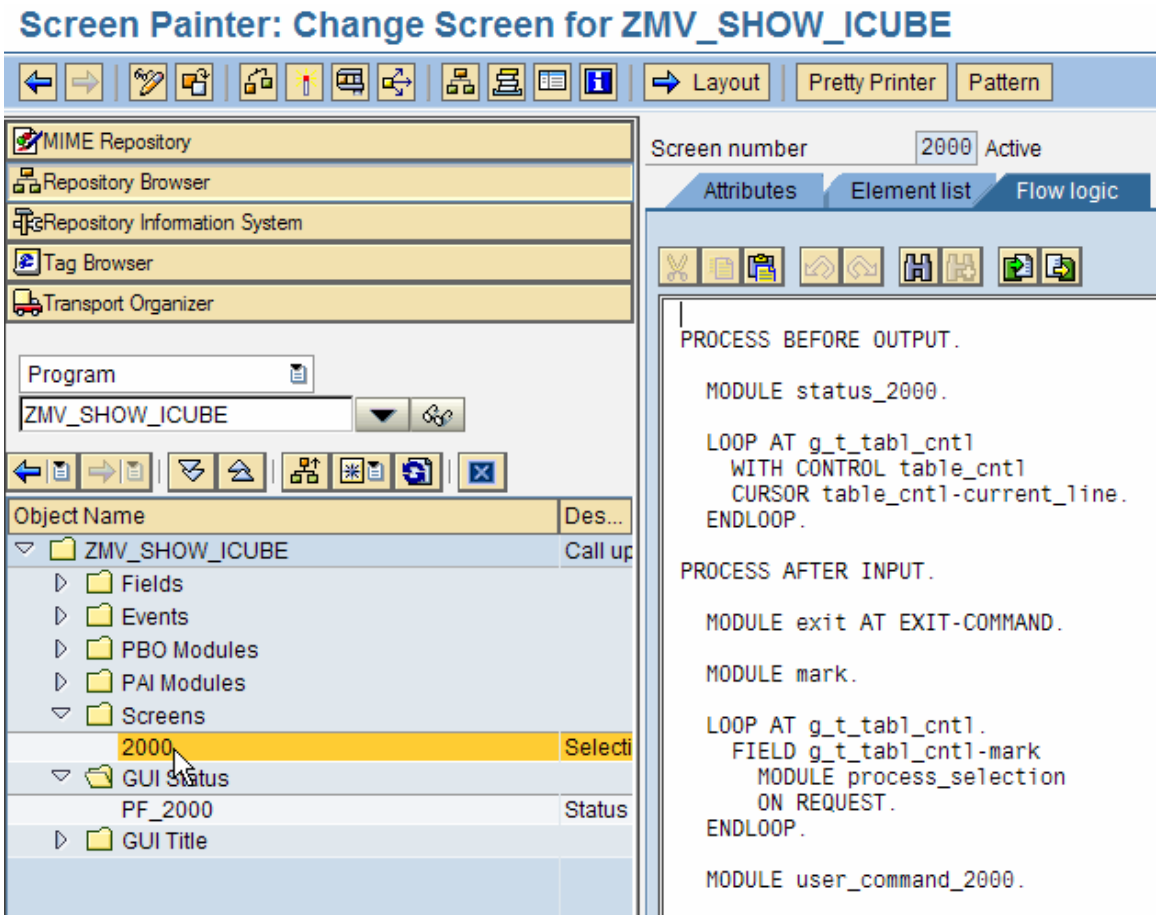
RETR → "Retrieve Selections"

The Application Toolbar will look as shown below at runtime

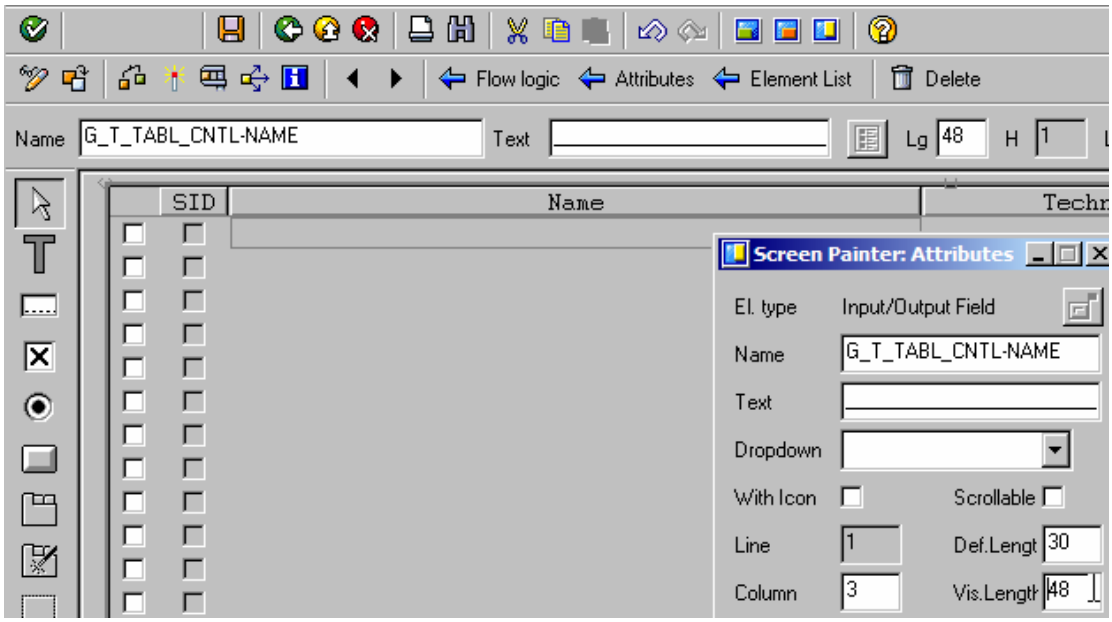


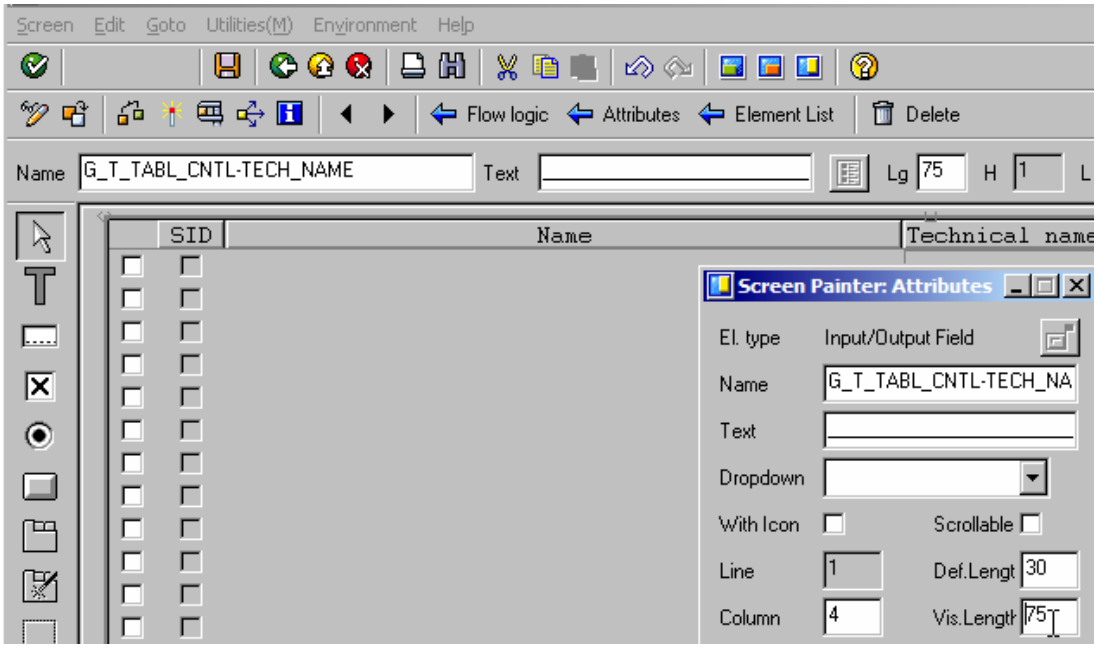

i. *Save* (Ctrl+S) and *Activate* (Ctrl+F3) *Status PF_2000*.

j. Now, Select *Screen 2000* for amendment.




k. Select Menu item, *Goto → Layout*. Now we need to make a minor modification to the Screen Layout to widen *Visible Length* of Columns with title *Name* and *Technical Name*. This can be done by changing the *Visible Length* of G_T_TABL_CNTL-NAME to 48 and that of G_T_TABL_CNTL-TECH_NAME to 75. (Please see the screen shots below)

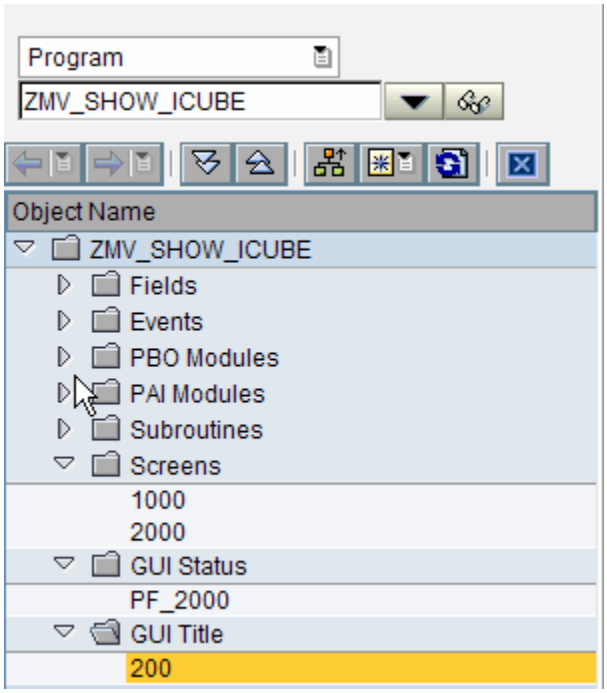




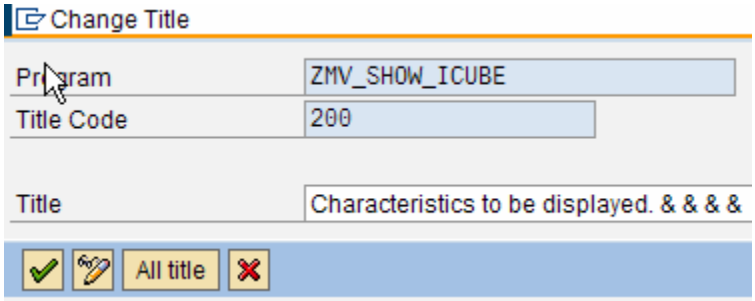

l. Click *Save.* Now Select Menu item, *Goto → Flow Logic*.

m. Change the Flow Logic to the code given in **Appendix A** titled *ABAP Code for Flow Logic* and Click *Save.*

n. Select *GUI Title 200* as shown below




o. Select *GUI Title 200* and change the Title to the one shown below. (This will be used by the tool to dynamically update the Title based on the number of objects selected.)

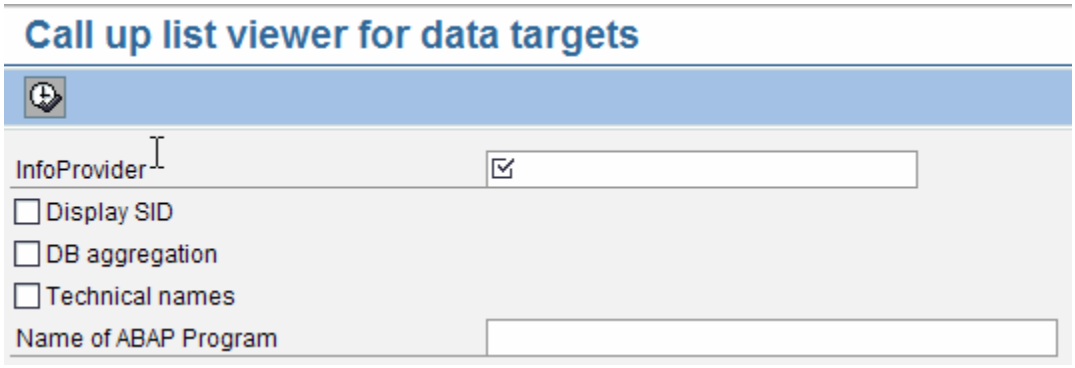



p. Activate Program ZMV_SHOW_ICUBE and dependent Objects. (Please see **Appendix 3** for adapting this code for BW 3.x. For further enhancement see Section **Suggested Enhancements** )

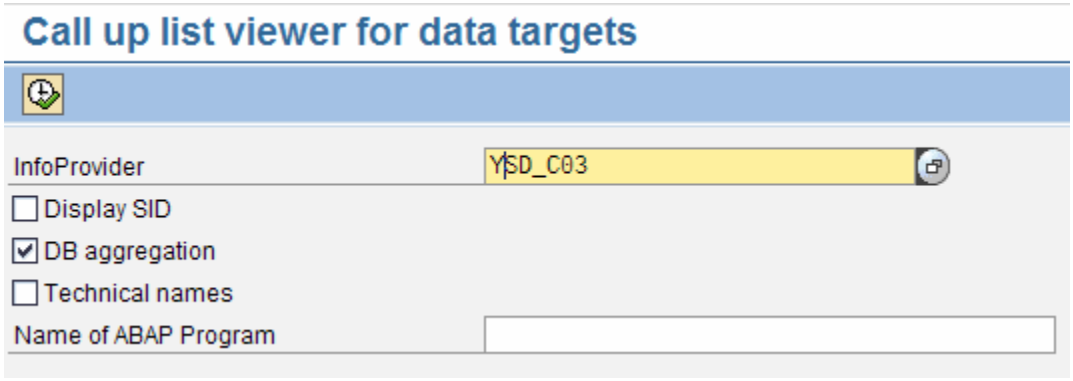
q. In SE93, create Transaction Code YLISTCUBE for program ZMV_SHOW_ICUBE. Now the Tool is ready to use.

## Usage Tips

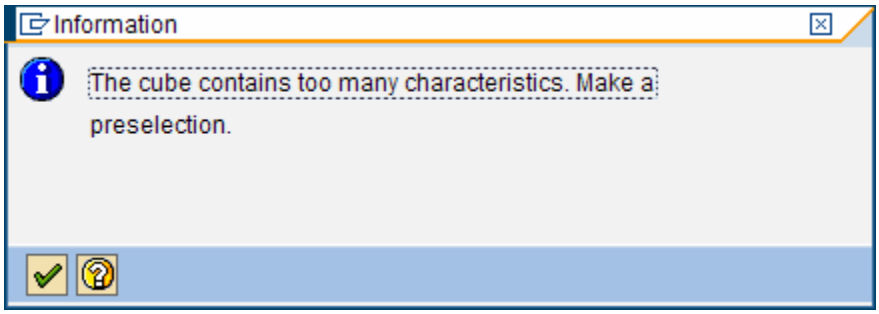
- This section provides tips for using this tool. The Infoprovider used for this example is an Infocube YSD_C03 that contains 154 Objects.
    a. Execute Program ZMV_SHOW_ICUBE (Use transaction code YLISTCUBE, if created). The following selection screen appears.



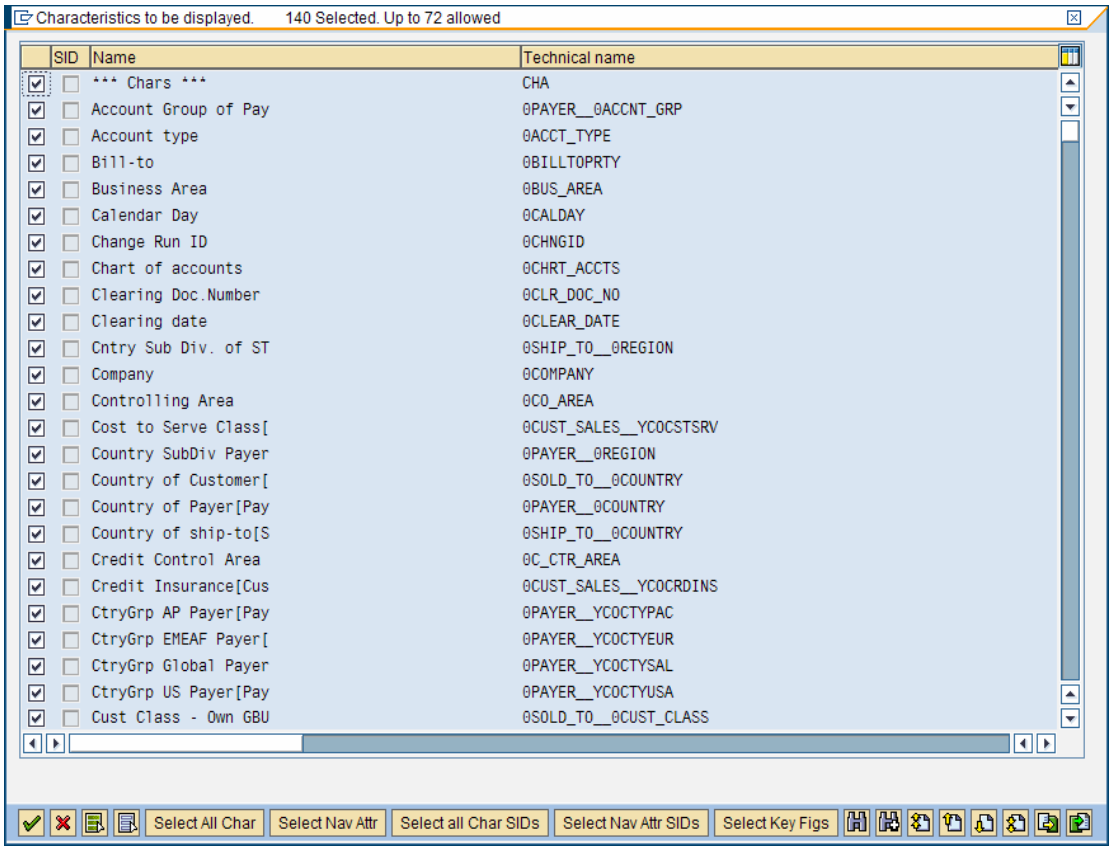

    b. Enter the Infoprovider name and Aggregation options.



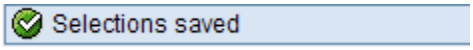

    c. Since the Infocube has many Characteristic objects, the following popup appears.



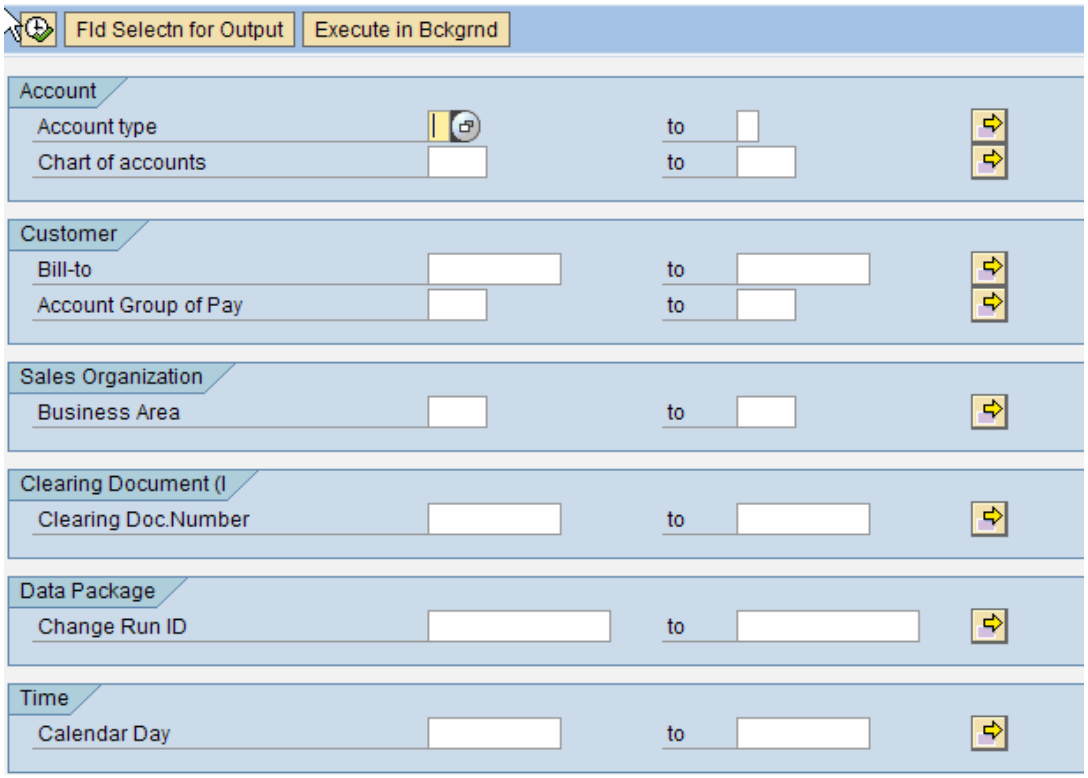

    d. A popup appears requesting user selection (*Screen 2000*). Notice that the Title bar shows the number of rows that are selected.



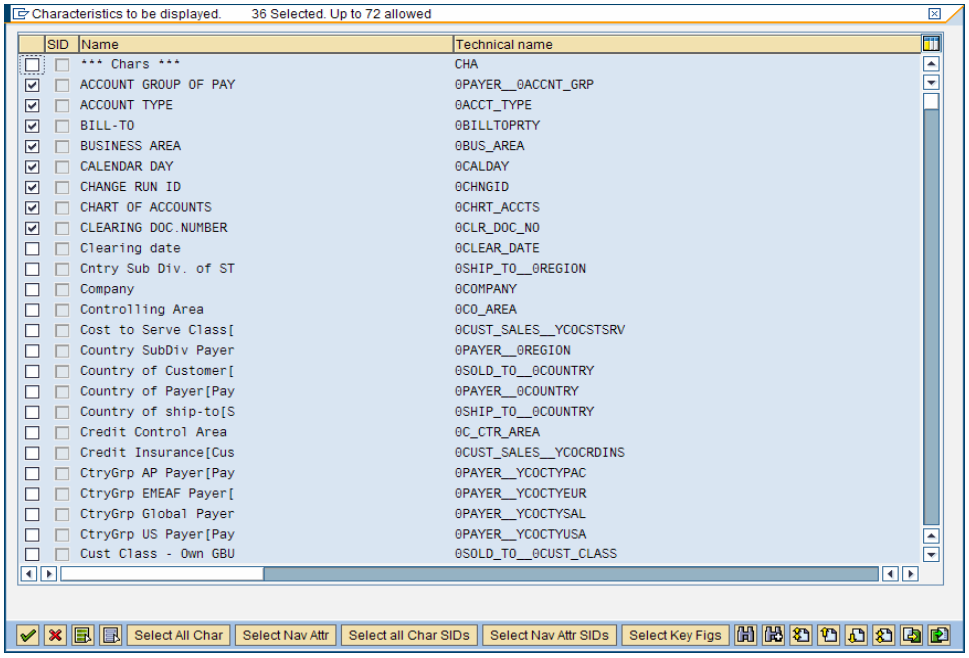

e. If you want to reuse your selections the next time you login, you can make the selection and select the icon  to *Store Selections for future use*. The following message appears if the selection was successfully saved. The selection is saved using the User Id and Infoprovider name.



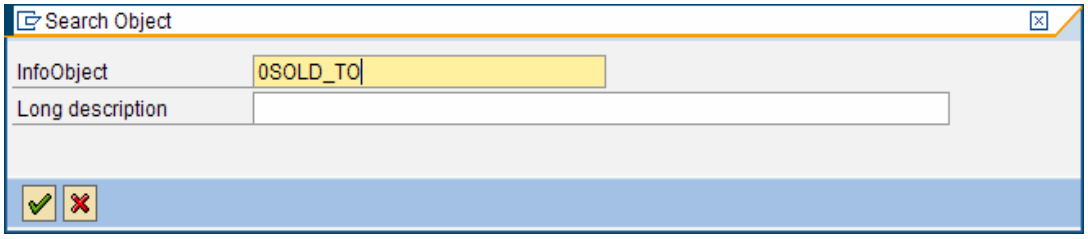

f. If you choose to Continue from this screen the generated selection screen appears (just as in the standard function.)




g. If you want to retrieve your saved selections the next time you execute the transaction, you can select the icon  to *Retrieve Past Selections*. The Checkboxes are automatically enabled based on your last saved selections.




h. You can use the Search Icons  and  to find an Object using its Description or Technical Name




i. You can also use the Page scrolling functions to navigate to the desired row.

## Suggested Enhancements

To further improve this tool according to your Company's or Project's needs you can do some of the following.

a. Choose to store the selection under a different key in INDX
b. For BW 3.x you can copy the relevant parts of the above code and incorporate it into a copied version of SAP Program RSDD_SHOW_ICUBE
c. Save the selections in such a way that it can be retrieved using a drop down menu
d. Cater to remodeling of Infocubes. TIP: Do not restore a Selection, if the Infocube was activated after the last save of the selections. (Use tables RSDCUBE and INDX with date option)
e. Option to free memory IDs periodically or when Infoprovider structure changes.

# Appendix A

## ABAP Code for the Flow Logic

This code should be used for the Flow Logic of Screen 2000 in program ZMV_SHOW_ICUBE.

```
<html>
  <head>
   <title>Custom LISTCUBE - ABAP Code for Flow Logic </title>
       <link rel="stylesheet" type="text/css" href="/sdn/css/csin.css"
       />
       <link rel="stylesheet" type="text/css" href="/sdn/css/sdn.css" />
  </head>
  <body>
     PROCESS BEFORE OUTPUT.
       MODULE status_2000.
       LOOP AT g_t_tabl_cntl
         WITH CONTROL table_cntl
         CURSOR table_cntl-current_line.
         module set_table.
       ENDLOOP.
     PROCESS AFTER INPUT.
       MODULE exit AT EXIT-COMMAND.
       LOOP AT g_t_tabl_cntl.
         FIELD g_t_tabl_cntl-mark
           MODULE process_selection
           ON REQUEST.
       ENDLOOP.
       MODULE mark.
       MODULE user_command_2000.
  </body>
</html>
```

## Appendix B

### ABAP Code for Program ZMV_SHOW_ICUBE

Given below is the code for program ZMV_SHOW_ICUBE. This is enhanced version of SAP standard program RSDD_SHOW_ICUBE.

```
<html>
  <head>
   <title>YLISTCUBE - ABAP Code for Flow Logic </title>
       <link rel="stylesheet" type="text/css" href="/sdn/css/csin.css"
       />
       <link rel="stylesheet" type="text/css" href="/sdn/css/sdn.css" />
  </head>
  <body>
       *&---------------------------------------------------------------------*
       *& Report  ZMV_SHOW_ICUBE                                              *
       *& Enhanced Tool that can be used in place of LISTCUBE                 *
       *&---------------------------------------------------------------------*
       *& This customized Version Authored by Manoj Vasudevan of XB5 Consulting
       *& October 2007.
       *&---------------------------------------------------------------------*
       REPORT  zmv_show_icube               .

       TYPE-POOLS: rsd, rsdq.

       DATA:
         g_r_dta          TYPE REF TO cl_rsd_dta,
         g_obj_overflow   TYPE rs_bool,
         g_okcode(4)      TYPE c,
         g_tlogo          TYPE rs_tlogo,
         g_t_tlogo        TYPE rs_t_tlogo,
         l_is_kyf         TYPE rs_bool,
         g_s_dta          TYPE rsd_s_dta,
         g_t_ioinf        TYPE rsdq_t_iobj_info,
         g_t_dta_dime     TYPE rsd_t_dta_dime,
         g_t_tabl_cntl    TYPE rsdq_t_tabl_cntl WITH HEADER LINE,
         g_t_dynp         LIKE dynpread OCCURS 0 WITH HEADER LINE.
       *-> INS MV
       data rc type c.
       data FIELDS like SVAL occurs 0 with header line.
       data tech_name like RSIOBJT-IOBJNM.
       data name like RSIOBJT-TXTLG.
       data cnt_marked like sy-tabix.
       data start_line  like sy-tabix.
       DATA  w_loopc like sy-loopc.
       *<- INS MV
```

```
CONTROLS:
  table_cntl       TYPE TABLEVIEW USING SCREEN 2000.

FIELD-SYMBOLS:
  <g_s_tabl_cntl>  TYPE rsdq_s_tabl_cntl.

PARAMETER:
  p_tlogo          TYPE rs_tlogo          NO-DISPLAY
                                          MEMORY ID /bic/rsdq/cubetype,
  p_dta            TYPE rsinfoprov        OBLIGATORY
                                          MEMORY ID /bic/rsdq/infoprov,
  p_ssids          TYPE dbman_show_sid    AS CHECKBOX
                                          MEMORY ID /bic/rsdq/showsids,
  p_sdims          TYPE dbman_show_dim    "AS  CHECKBOX
                                          "MEMORY ID /bic/rsdq/showdims
                                          NO-DISPLAY,
  p_dbagg          TYPE dbman_db_aggregation
                                          AS CHECKBOX
                                          MEMORY ID /bic/rsdq/db_aggr,
  p_tchnm          TYPE dbman_technm      AS CHECKBOX
                                          MEMORY ID /bic/rsdq/tech_nms,
  p_repnm          TYPE rsdr0_repnm.

SET PARAMETER ID '/BIC/RSDQ/INFOPROV'   FIELD p_dta.
SET PARAMETER ID '/BIC/RSDQ/SHOWSIDS'   FIELD p_ssids.
SET PARAMETER ID '/BIC/RSDQ/SHOWDIMS'   FIELD p_sdims.
SET PARAMETER ID '/BIC/RSDQ/DB_AGGR'    FIELD p_dbagg.
SET PARAMETER ID '/BIC/RSDQ/TECH_NMS'   FIELD p_tchnm.
SET PARAMETER ID '/BIC/RSDQ/REPNM'      FIELD p_repnm.

* Displaying DIMIDS is not implemented yet
p_sdims = rs_c_false.

* retrieve information about the infocube infoobjects
CALL FUNCTION 'RSDQ_GET_DTA_INFO'
  EXPORTING
    i_infoprov   = p_dta
    i_show_sids  = p_ssids
  IMPORTING
    e_s_dta      = g_s_dta
    e_t_ioinf    = g_t_ioinf
    e_t_dta_dime = g_t_dta_dime
  EXCEPTIONS
    OTHERS       = 1.
IF sy-subrc <> 0.
  MESSAGE e203(dbman) WITH p_dta.
```

```
ENDIF.

SET PARAMETER ID '/BIC/RSDQ/CUBETYPE'   FIELD g_s_dta-tlogo.

DO.
* check whether all infoobjects fit into the
* report to be generated
  CALL FUNCTION 'RSDQ_TABLCNTL_FROM_IOINF'
    EXPORTING
      i_t_ioinf      = g_t_ioinf
      i_called_from  = 'L'         "Called by Listcube
    IMPORTING
      e_t_tabl_cntl  = g_t_tabl_cntl[]
      e_obj_overflow = g_obj_overflow.

  IF g_obj_overflow EQ rs_c_true.
*   customer has to preselect infoobjects to be
*   displayed in selection screen of report
    MESSAGE i201(rsdd).

    l_is_kyf = rs_c_false.

*   In preselection panel clear the marks for all
*   infoobjects except for the key figures ...
    LOOP AT g_t_tabl_cntl[]
      ASSIGNING <g_s_tabl_cntl>.

      IF <g_s_tabl_cntl>-tech_name = 'KYF'.
        l_is_kyf = rs_c_true.
      ENDIF.

      IF l_is_kyf = rs_c_false.
        CLEAR <g_s_tabl_cntl>-mark.
      ENDIF.
    ENDLOOP.

*   display preselection panel
    CALL SCREEN 2000
         STARTING AT 10 10.

*   retrieve infoobjects to be displayed.
    CALL FUNCTION 'RSDQ_IOINF_FROM_TABLCNTL'
      EXPORTING
        i_t_tabl_cntl = g_t_tabl_cntl[]
      CHANGING
        c_t_ioinf     = g_t_ioinf.
```

```
  ELSE.
    EXIT.
  ENDIF.
ENDDO.

* build the infocube-specific listreport and
* execute it.
CALL FUNCTION 'RSDQ_VIEW_INFOPROV'
  EXPORTING
    i_infoprov           = p_dta
    i_s_dta              = g_s_dta
    i_t_dta_dime         = g_t_dta_dime
    i_show_sids          = p_ssids
    i_show_dimids        = p_sdims
    i_tech_nms           = p_tchnm
    i_use_db_aggregation = p_dbagg
    i_repnm              = p_repnm
  CHANGING
    c_t_ioinf            = g_t_ioinf
  EXCEPTIONS
    dta_not_found        = 1
    illegal_input        = 2
    OTHERS               = 4.
IF sy-subrc EQ 1.
  MESSAGE i151(brain) WITH p_dta.
ELSEIF sy-subrc > 1.
*-> DEL MV
*  MESSAGE i299(brain) WITH 'LISTCUBE' 'RSDQ_VIEW_DATATARGET'.
*<- DEL MV
*-> INS MV
  MESSAGE i299(brain) WITH 'LISTCUBE' 'RSDQ_VIEW_INFOPROV'.
*<- INS MV
ENDIF.

*******************************
* end of report
*******************************
AT SELECTION-SCREEN ON VALUE-REQUEST FOR p_dta.

* g_t_dynp-fieldname = 'P_CUBET'.
*  g_t_dynp-fieldname = 'P_TLOGO'. "UK 998954
*  APPEND g_t_dynp.
  g_t_dynp-fieldname = 'P_DTA'.
  APPEND g_t_dynp.

  CALL FUNCTION 'DYNP_VALUES_READ'
```

```
    EXPORTING
      dyname               = 'RSDD_SHOW_ICUBE'
      dynumb               = '1000'
      translate_to_upper   = 'X'
    TABLES
      dynpfields           = g_t_dynp
    EXCEPTIONS
      invalid_abapworkarea = 1
      invalid_dynprofield  = 2
      invalid_dynproname   = 3
      invalid_dynpronummer = 4
      invalid_request      = 5
      no_fielddescription  = 6
      invalid_parameter    = 7
      undefind_error       = 8
      OTHERS               = 9.

*  CHECK sy-subrc = 0.

  READ TABLE g_t_dynp
       WITH KEY fieldname = 'P_DTA'.
  p_dta = g_t_dynp-fieldvalue.

  CLEAR g_t_tlogo.
  READ TABLE g_t_dynp
       WITH KEY fieldname = 'P_TLOGO'.

  p_tlogo = g_t_dynp-fieldvalue.
  IF p_tlogo IS INITIAL OR
     ( p_tlogo <> rs_c_tlogo-infocube      AND
       p_tlogo <> rs_c_tlogo-aggregate     AND
       p_tlogo <> rs_c_tlogo-multiprovider AND
       p_tlogo <> rs_c_tlogo-ods_object    AND
       p_tlogo <> rs_c_tlogo-infoobject    AND
       p_tlogo <> rs_c_tlogo-infoset ).
    CLEAR p_tlogo.
    CLEAR g_t_tlogo.
    APPEND rs_c_tlogo-infocube      TO g_t_tlogo.
    APPEND rs_c_tlogo-aggregate     TO g_t_tlogo.
    APPEND rs_c_tlogo-multiprovider TO g_t_tlogo.
    APPEND rs_c_tlogo-ods_object    TO g_t_tlogo.
    APPEND rs_c_tlogo-infoobject    TO g_t_tlogo.
    APPEND rs_c_tlogo-aggrlevel     TO g_t_tlogo.
    APPEND rs_c_tlogo-infoset       TO g_t_tlogo.
  ENDIF.
```

```
  CALL METHOD cl_rsd_dta=>f4
    EXPORTING
      i_tlogo    = p_tlogo
      i_t_tlogo  = g_t_tlogo
    CHANGING
      c_infoprov = p_dta.

*&---------------------------------------------------------------------*
*&      Module  PROCESS_SELECTION  INPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE process_selection INPUT.

  MODIFY g_t_tabl_cntl INDEX table_cntl-current_line.

ENDMODULE.                             " PROCESS_SELECTION  INPUT

*&---------------------------------------------------------------------*
*&      Module  STATUS_2000  OUTPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE status_2000 OUTPUT.
  SET PF-STATUS 'PF_2000'.
*-> DEL MV
*  SET TITLEBAR '200'.
*<- DEL MV
*-> INS MV
  clear cnt_marked.
  loop at g_t_tabl_cntl where mark = 'X'.
    cnt_marked = cnt_marked + 1.
  endloop.
  SET TITLEBAR '200' with cnt_marked 'Selected. Up to 72 allowed'.
*<- INS MV
  CLEAR g_okcode.
ENDMODULE.                             " STATUS_2000  OUTPUT

*&---------------------------------------------------------------------*
*&      Module  EXIT  INPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE exit INPUT.
  IF g_okcode EQ 'CNCL'.
    LEAVE PROGRAM.
```

```
  ENDIF.
ENDMODULE.                              " EXIT  INPUT

*&---------------------------------------------------------------------*
*&      Module  MARK  INPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE mark INPUT.
*-> INS MV
  DATA p_mid TYPE MEMORY_ID.
*<- INS MV
  FIELD-SYMBOLS:
    <l_s_tabl_cntl>  TYPE rsdq_s_tabl_cntl.

  CASE g_okcode.
    WHEN 'MALL'.
      LOOP AT g_t_tabl_cntl ASSIGNING <l_s_tabl_cntl>.
        <l_s_tabl_cntl>-mark = rs_c_true.
      ENDLOOP.

    WHEN 'DALL'.
      LOOP AT g_t_tabl_cntl ASSIGNING <l_s_tabl_cntl>.
        <l_s_tabl_cntl>-mark = rs_c_false.
      ENDLOOP.

    WHEN 'MCHA'.
      LOOP AT g_t_tabl_cntl ASSIGNING <l_s_tabl_cntl>.
        IF <l_s_tabl_cntl>-tech_name NP '*__*'
          AND <l_s_tabl_cntl>-sid EQ rs_c_false.
          <l_s_tabl_cntl>-mark = rs_c_true.
        ENDIF.
      ENDLOOP.

    WHEN 'MSID'.
      LOOP AT g_t_tabl_cntl ASSIGNING <l_s_tabl_cntl>.
        IF <l_s_tabl_cntl>-tech_name NP '*__*'
          AND <l_s_tabl_cntl>-sid EQ rs_c_true.
          <l_s_tabl_cntl>-mark = rs_c_true.
        ENDIF.
      ENDLOOP.

    WHEN 'NKEY'.
      LOOP AT g_t_tabl_cntl ASSIGNING <l_s_tabl_cntl>.
        IF <l_s_tabl_cntl>-tech_name CP '*__*'
          AND <l_s_tabl_cntl>-sid EQ rs_c_false.
```

```
            <l_s_tabl_cntl>-mark = rs_c_true.
          ENDIF.
        ENDLOOP.

      WHEN 'NSID'.
        LOOP AT g_t_tabl_cntl ASSIGNING <l_s_tabl_cntl>.
          IF <l_s_tabl_cntl>-tech_name CP '*__*'
            AND <l_s_tabl_cntl>-sid EQ rs_c_true.
            <l_s_tabl_cntl>-mark = rs_c_true.
          ENDIF.
        ENDLOOP.

      WHEN 'KYF'.
        l_is_kyf = rs_c_false.

*     clear the marks except for all key figures ...
        LOOP AT g_t_tabl_cntl[]
          ASSIGNING <l_s_tabl_cntl>.

          IF <l_s_tabl_cntl>-tech_name = 'KYF'.
            l_is_kyf = rs_c_true.
          ENDIF.

          IF l_is_kyf = rs_c_true.
            <l_s_tabl_cntl>-mark = rs_c_true.
          ENDIF.
        ENDLOOP.
*-> INS MV
      WHEN 'FIND' or 'FINN'.
        IF g_okcode = 'FIND' or FIELDS[] is initial.
          CLEAR: FIELDS[], FIELDS.
     FIELDS-TABNAME = 'RSIOBJT'.FIELDS-FIELDNAME = 'IOBJNM'.APPEND FIELDS.
     FIELDS-TABNAME = 'RSIOBJT'.FIELDS-FIELDNAME = 'TXTLG' .APPEND FIELDS.
          CALL FUNCTION 'POPUP_GET_VALUES'
            EXPORTING
              NO_VALUE_CHECK  = 'X'
              POPUP_TITLE     = 'Search Object'
              START_COLUMN    = '10'
              START_ROW       = '20'
            IMPORTING
              RETURNCODE      = rc
            TABLES
              FIELDS          = fields
            EXCEPTIONS
              ERROR_IN_FIELDS = 1
              OTHERS          = 2.
```

```
*
      endif.
      clear: name, tech_name.
      read table fields with key FIELDNAME = 'IOBJNM'.
      tech_name = fields-value.
      read table fields with key FIELDNAME = 'TXTLG'.
      name = fields-value.
      start_line = table_cntl-top_line + 1.
      LOOP AT g_t_tabl_cntl .
        SEARCH g_t_tabl_cntl for tech_name starting at start_line.
        if sy-subrc = 0.
          table_cntl-top_line = sy-tabix . exit.
        endif.
        SEARCH g_t_tabl_cntl for name starting at start_line.
        if sy-subrc = 0.
          table_cntl-top_line  = sy-tabix . exit.
        endif.
      ENDLOOP.
    when 'P--' or                    "top of list
         'P-'  or                    "previous page
         'P+'  or                    "next page
         'P++'.                      "bottom of list
      perform SCROLL using g_okcode. LEAVE SCREEN.
    WHEN 'STOR'. "Store
      Concatenate p_dta  sy-uname into p_mid.
      EXPORT g_t_tabl_cntl TO DATABASE INDX(ST) ID P_MID .
      if sy-subrc <> 0.
        Message S368(00) with 'Error while storing'.
      Else.
        Message S368(00) with 'Selections saved'.
      endif.
    WHEN 'RETR'. "Retrieve
      Concatenate p_dta  sy-uname into p_mid.
      IMPORT g_t_tabl_cntl FROM DATABASE INDX(ST) ID P_MID .
      if sy-subrc <> 0.
        Message S368(00) with 'Nothing stored'.
      Else.
        Message S368(00) with 'Saved selections retrieved'.
      endif.
*<- INS MV
    WHEN OTHERS.
  ENDCASE.
ENDMODULE.                             " MARK  INPUT

*&---------------------------------------------------------------------*
*&      Module  USER_COMMAND_2000  INPUT
```

```
*&---------------------------------------------------------------------*
MODULE user_command_2000 INPUT.
  IF g_okcode EQ 'OK'.
    READ TABLE g_t_tabl_cntl
      WITH KEY mark = rs_c_true
      TRANSPORTING NO FIELDS.

    IF sy-subrc <> 0.
      MESSAGE i203(rsdd).
      RETURN.
    ENDIF.

    SET SCREEN 0.
    LEAVE SCREEN.
  ENDIF.
ENDMODULE.                             " USER_COMMAND_2000  INPUT

*-> INS MV
*&---------------------------------------------------------------------*
*&      Form  SCROLL
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->P_OK       text
*----------------------------------------------------------------------*
form SCROLL using p_ok .
*-BEGIN OF LOCAL DATA-------------------------------------------------*
  data l_tc_new_top_line     type i.
  data l_tc_name             like feld-name.
  data l_tc_lines_name       like feld-name.
  data l_tc_field_name       like feld-name.

  field-symbols <tc>         type cxtab_control.
  field-symbols <lines>      type i.
*-END OF LOCAL DATA---------------------------------------------------*

*  assign (p_tc_name) to <tc>.
* get looplines of TableControl
*  concatenate 'G_' p_tc_name '_LINES' into l_tc_lines_name.
*  assign (l_tc_lines_name) to <lines>.

* is no line filled?                                                  *
  if TABLE_CNTL-lines = 0.
*   yes, ...                                                         *
    l_tc_new_top_line = 1.
  else.
```

```
*   no, ...                                                          *
    call function 'SCROLLING_IN_TABLE'
         exporting
              entry_act             = TABLE_CNTL-top_line
              entry_from            = 1
              entry_to              = TABLE_CNTL-lines
              last_page_full        = 'X'
              loops                 = w_loopc
              ok_code               = p_ok
              overlapping           = 'X'
         importing
              entry_new             = l_tc_new_top_line
         exceptions
*             NO_ENTRY_OR_PAGE_ACT  = 01
*             NO_ENTRY_TO           = 02
*             NO_OK_CODE_OR_PAGE_GO = 03
              others                = 0.
  endif.

**get actual tc and column                                           *
*  get cursor field l_tc_field_name
*             area  l_tc_name.
*
*  if syst-subrc = 0.
**    set actual column                                              *
*      set cursor field l_tc_field_name line 1.
*  endif.

* set the new top line                                               *
  TABLE_CNTL-top_line = l_tc_new_top_line.
endform.                               "
*&---------------------------------------------------------------------*
*&      Module  set_table  OUTPUT
*&---------------------------------------------------------------------*
MODULE set_table OUTPUT.
  w_loopc = sy-loopc.
ENDMODULE.                 " set_table  OUTPUT
*<- INS MV
</body>
</html>
```

# Appendix C

## Adapting the Code for BW 3.x

Following are the guidelines for adapting the above code for BW 3.x. Please contact me if you need the complete code. The code that references the new functions codes in GUI Status PF_2000 and the selection screen field p_repnm should also be commented out.

- **Comment out the following lines of code**

```
p_repnm          TYPE rsdr0_repnm.

….

SET PARAMETER ID '/BIC/RSDQ/REPNM'      FIELD p_repnm.

….

APPEND rs_c_tlogo-aggrlevel     TO g_t_tlogo.
```

- **Replace the following lines of code …….**

```
CALL FUNCTION 'RSDQ_VIEW_INFOPROV'
  EXPORTING
    i_infoprov           = p_dta
    i_s_dta              = g_s_dta
    i_t_dta_dime         = g_t_dta_dime
    i_show_sids          = p_ssids
    i_show_dimids        = p_sdims
    i_tech_nms           = p_tchnm
    i_use_db_aggregation = p_dbagg
    i_repnm              = p_repnm
  CHANGING
    c_t_ioinf            = g_t_ioinf
  EXCEPTIONS
    dta_not_found        = 1
    illegal_input        = 2
    OTHERS               = 4.
IF sy-subrc EQ 1.
  MESSAGE i151(brain) WITH p_dta.
ELSEIF sy-subrc > 1.
*-> DEL MV
*  MESSAGE i299(brain) WITH 'LISTCUBE' 'RSDQ_VIEW_DATATARGET'.
*<- DEL MV
*-> INS MV
  MESSAGE i299(brain) WITH 'LISTCUBE' 'RSDQ_VIEW_INFOPROV'.
*<- INS MV
ENDIF.
```

**…. with the following lines of code below**

```
CALL FUNCTION 'RSDQ_VIEW_DATATARGET'
  EXPORTING
    i_datatarget         = p_dta
    i_s_dta              = g_s_dta
    i_t_dta_dime         = g_t_dta_dime
    i_show_sids          = p_ssids
    i_show_dimids        = p_sdims
    i_tech_nms           = p_tchnm
    i_use_db_aggregation = p_dbagg
  CHANGING
    c_t_ioinf            = g_t_ioinf
  EXCEPTIONS
    dta_not_found        = 1
    illegal_input        = 2
    OTHERS               = 4.
IF sy-subrc EQ 1.
  MESSAGE i151(brain) WITH p_dta.
```

```
ELSEIF sy-subrc > 1.
  MESSAGE i299(brain) WITH 'LISTCUBE' 'RSDQ_VIEW_DATATARGET'.
ENDIF.
```

- **Replace the following lines of code …….**

```
CALL FUNCTION 'RSDQ_GET_DTA_INFO'
   EXPORTING
     i_infoprov   = p_dta
```

**…. with the following lines of code below**

```
CALL FUNCTION 'RSDQ_GET_DTA_INFO'
  EXPORTING
    i_datatarget = p_dta
```

## Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.